Proyecto Fin de Máster

Máster en Ingeniería de Telecomunicación

Red de sensores para localización basada en LoRa y FIWARE

Autora: Gloria Martínez Muñoz
Tutor: Antonio Jesús Sierra Collado

Dpto. de Ingeniería Telemática

Escuela Técnica Superior de Ingeniería

Sevilla, 2019

Proyecto Fin de Máster

Máster en Ingeniería de Telecomunicación

Red de sensores para localización basada en LoRa y FIWARE

Autora:
Gloria Martínez Muñoz

Tutor:
Antonio Jesús Sierra Collado

Dpto. de Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2019

Proyecto Fin de Máster: Red de sensores para localización basada en LoRa y FIWARE

Autora: Gloria Martínez Muñoz
Tutor: Antonio Jesús Sierra Collado

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:
Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2019

El Secretario del Tribunal

A mi familia

Agradecimientos

Quiero agradecer a todas las personas que han contribuido de una u otra manera a la llegada de este momento en el que pongo fin a mis estudios de máster. Tanto profesores como compañeros y amigos han aportado su granito de arena y han ayudado a que esta etapa sea más llevadera y termine con muchas anécdotas que recordar.

Gracias, papá y mamá, por levantarme y apoyarme en los momentos más difíciles. El final de esta etapa habría sido imposible sin vuestra atención diaria; mis triunfos son vuestros también. Gracias abuelita, por fin estás viendo cómo pongo este broche final por el que tanto preguntabas.

Gloria Martínez Muñoz
Sevilla, 2019

Resumen

La comunicación y conexión de los diferentes objetos que nos rodean es uno de los principales objetivos que actualmente ocupan a la industria tecnológica. Esto es lo que hoy se conoce como Internet de las Cosas (IoT por sus siglas en inglés). Una consecuencia directa de esta tendencia es que una gran cantidad de datos deberán ser gestionados y procesados con el objetivo de poder sintetizar y seleccionar la información requerida por un usuario o sistema que quiera hacer uso de esa información.

Este escenario plantea la necesidad de poder acceder a los datos de una forma sencilla sin necesidad de conocer el protocolo nativo que se emplea para la recolección de esos datos o donde se encuentran. Esta necesidad se ve agravada por la heterogeneidad que se da en este tipo de entornos, así como por la simplicidad de los objetos que generan estos datos, incapaces de soportar complejos protocolos.

Sería deseable, por tanto, ser capaz de obtener de manera homogénea y agnóstica a la complejidad subyacente la información necesaria en el momento que se necesite o solicite. Es por ello que resulta obligatorio en el desarrollo de cualquier escenario de estas características contar con un intermediario entre los productores de contexto (objetos, sensores, etc.) y las aplicaciones consumidoras de contexto.

En este proyecto se ha abordado esta problemática integrando un sistema de gestión de recursos e información de contexto, desarrollado a partir de un conjunto de componentes disponibles dentro de la plataforma FIWARE.

Los datos gestionados en esta plataforma serán generados por una red de sensores. En los últimos años se han producido grandes avances en el desarrollo de este tipo de sistemas: comunicaciones, organización de arquitecturas, orquestación de servicios, seguridad, etc. Desde la aparición de Internet, la sociedad ha cambiado sus formas de comunicarse e interrelacionarse. El Internet de las cosas está cambiando la forma en que nos relacionamos y en la que se relacionan los objetos y dispositivos de los que nos rodeamos a diario.

Existen multitud de tecnologías utilizadas para la interconexión de estos objetos, para su control y gestión y para transportar la información que generan. En lo que se refiere a la arquitectura de red, han surgido numerosos modelos y tecnologías que proponen acercar las capacidades de procesamiento y almacenamiento a los nodos finales de la red. En cuanto a las tecnologías de comunicación utilizadas, existen diversas opciones según la aplicación final que reciba la red.

En este proyecto también se introducen algunas de las numerosas tecnologías de comunicación surgidas para el desarrollo e implementación de redes de sensores, focalizando el estudio en la tecnología de comunicación inalámbrica LoRa.

Se expone un estudio sobre las capacidades y limitaciones de este protocolo, incluyendo la descripción del proceso de implementación de una red de sensores en un entorno real empleando esta tecnología, y detallando las herramientas, el software y la configuración utilizadas para el correcto funcionamiento de una red de este tipo. Se mostrarán los resultados obtenidos tras realizar algunas pruebas para comprobar las prestaciones de la tecnología en cuestión.

Abstract

The communication and connectivity of the different objects that surround us is one of the main objectives that are being addressed by the technological industry nowadays. Today, this is known as the Internet of Things (IoT). A direct consequence of this concept is that a huge quantity of data must be managed and processed so as to synthesize and select the information required by a user or system that intends to make use of it.

This scenario poses the need of being able to access the data in an easy way, without need of knowing the native protocol being used for collecting this data, or its whereabouts. This need is even more challenging considering the heterogeneity given in this kind of environments, as well as the simplicity of the objects that generate that data, unable to cope with complex protocols.

Therefore, it would be desirable being able to obtain the necessary information, in a homogeneous and way agnostic to the underlying complexity, in the moment that it is needed or requested. Consequently, in the development of any scenario of these characteristics, it is mandatory to count with an intermediary between the context producers (objects, sensors, etc) and the applications that consume this context information.

These challenges have been tackled in this project by integrating a resource and information context management system for the IoT. This system has been developed by integrating a collection of generic enablers within the FIWARE platform.

The data managed in this platform will be generated by a sensor network. In the last years there have been great advances in the development of this type of systems: communications, organization of architectures, orchestration of services, security... From the beginning of the Internet, society has changed the ways of communication and interrelated. The Internet of Things is changing the way we relate to each other and the objects and devices that surround us every day.

There are many technologies used for the interconnection of these objects, for their control and management and to transport the information they generate. With regard to the network architecture, numerous models and technologies have emerged that propose bringing the processing and storage capacities closer to the final nodes of the network. Regarding the communication technologies used, there are several options according to the final application received by the network.

This project also introduces some of the numerous communication technologies that have arisen for the development and implementation of sensor networks, focusing the study on the LoRa wireless communication technology.

A study on the capabilities and limitations of this protocol is presented in this document, including the description of the process of implementing a sensor network in a real environment using this technology, and detailing the tools, software and configuration used for the proper functioning of a network of this type. The results obtained after performing some tests to check the performance of the technology in question will be explained.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

1.1 Motivación y objetivos

En los últimos años se han desarrollado tecnologías que permiten la conexión y comunicación de distintos objetos que nos rodean. Mediante su uso se generan enormes volúmenes de datos que pueden llegar a ser difíciles de manejar sin las herramientas adecuadas.

En este proyecto se desarrolla una aplicación basada en FIWARE, una plataforma de gestión de datos de contexto. Además, se implementará una red de sensores basada en la tecnología inalámbrica LoRa, además de realizar un estudio sobre la misma.

El objetivo de este proyecto es, por tanto, experimentar con la tecnología LoRa y utilizarla para diseñar la red de sensores que interactúe con un gestor de información de Contexto. Los datos almacenados y gestionados por éste podrán ser visualizados a través de una sencilla aplicación web. El fin de este proyecto es la experimentación con ambas tecnologías: LoRa y FIWARE.

1.2 Estructura del trabajo de fin de Máster

A continuación, se resume el contenido de los diferentes capítulos y anexos que componen este proyecto.

- **Capítulo 1: Introducción.**

  Se trata del capítulo actual, en el cual se resumen los motivos que han propiciado el desarrollo de este proyecto y los objetivos que se persiguen con su realización.

- **Capítulo 2: Estado del Arte.**

  En este capítulo se describen conceptos esenciales que se deben conocer y tener en cuenta para comprender este documento y los conceptos que se trabajarán en el proyecto.

- **Capítulo 3: La tecnología LoRa.**

  Este capítulo profundiza y expone la tecnología de comunicación fundamental que se desarrolla y utiliza en el proyecto, y que permite la comunicación entre los dispositivos que componen la red desarrollada.

- **Capítulo 4: La plataforma Fiware.**

  En este capítulo se explica en qué consiste la plataforma Fiware y se detallan conceptos fundamentales para este proyecto. Concretamente se centrará en el elemento Context Broker, que será parte fundamental en el desarrollo del mismo.

- **Capítulo 5: Arquitectura y componentes.**

En esta sección se detallan el escenario de aplicación de este proyecyo y los elementos hardware y fiware empleados en la realización del mismo, así como sus caracterísiticas y funcionalidades.

- **Capítulo 6: Software e implementación.**

  En este capítulo se muestra en detalle el funcionamiento de los componentes empleados y su configuración. Se detalla la solución a la que se ha llegado, la red construída y la programación implementada. Asímismo, se explica cómo se han realizado las conexiones y cómo se muestran los datos a través de una aplicación web.

- **Capítulo 7: Pruebas y validaciones.**

  En esta sección se detallan las pruebas de funcionamiento realizadas y los resultados obtenidos a través de la experimentación y la puesta en marcha de la red construída. Se comentan los problemas encontrados durante la realización del proyecto.

- **Capítulo 8: Conclusiones y líneas futuras.**

  En este último capítulo se detallan las conclusiones alcanzadas con el desarrollo del proyecto y las líneas futuras del mismo.

2 ESTADO DEL ARTE

En este documento se tratan aspectos variados, los cuales no pueden ser comprendidos sin antes introducir una base teórica sobre ellos.

A fin de comprender la situación tecnológica actual de la que se parte, en este capítulo van a tratarse los siguientes conceptos:

- El Internet de las Cosas
- Tecnologías de comunicación en el Internet de las Cosas

2.1 El internet de las Cosas

El Internet de las Cosas o *Internet of Things* [1], en inglés, por cuyas siglas suele ser referido normalmente como IoT, es una idea basada en la existencia de una capa de conectividad digital para cosas existentes. Con el término "cosas" nos referimos a cualquier participante real o virtual, ya sean objetos cotidianos, seres humanos, datos virtuales o agentes de software inteligente.

Se trata de la construcción de un mundo en el que cada objeto tiene una identidad virtual propia y capacidad potencial para integrarse e interactuar de manera independiente en la red con cualquier otro individuo, ya sea una máquina o un humano. El objetivo que se persigue es crear un entorno en el que la infomación básica proveniente de cualquiera de los actores autónomos conectados pueda ser compartida eficientemente con otros, en tiempo real.

Ilustración 1: Internet de las Cosas

De esta forma, los objetos pasan a tener un papel activo en el mundo real y adquieren la capacidad de responder frente a la ocurrencia de un determinado evento, desencadenando algún tipo de acción.

Con la aparición del Internet de las Cosas, se está llevando a cabo una transformación en la que se están creando nuevos modelos de negocio, productos y compañías. Se espera que esta idea traiga consigo (y ya lo está empezando a hacer) numerosos beneficios en aspectos como la optimización de la cadena de abastecimiento, efectividad de costos, mejoras en la experiencia de consumidores, beneficios en aspectos de seguridad y servicios de emergencia, etc.

2.1.1 Historia del Internet de las Cosas

El término "Internet de las Cosas" fue acuñado por primera vez en el año 2009 por el profesor del Massachusetts Institute of Technology Kevin Ashton, durante una conferencia. Si bien es cierto que el IoT como concepto surgió unos años antes.

A lo largo del siglo XX se realizaron numerosos experimentos relacionados con el mundo de las telecomunicaciones, para tratar de conectar diversos aparatos entre sí. En las primeras décadas del siglo algunos científicos ya soñaban con la idea de conectar objetos y que éstos trabajaran utilizando inteligencia propia, como Alan Turing o Nikola Tesla. Sin embargo, la inmadurez tecnológica de entonces les impidió profundizar en el tema.

Entre los años 60 y 70 nació la red ARPENET, que más tarde se convertiría en el actual Internet. Pero en sus primeros años solo se utilizaba con fines militares y académicos, por lo que su uso estaba restringido. No fue hasta mediado de los años 90, con la globalización del Internet y el avance y desarrollo de los protocolos, cuando se empezó a trabajar más seriamente con la idea de interconectar objetos.

Tras la alusión al término por primera vez en aquella conferencia, muchos investigadores comenzaron a trabajar en el Internet de las Cosas. En el año 2008, un grupo de empresas crearon la IPSO Alliance (Internet Protocol for Smart Objects), una alianza para desarrollar e invertigar el Internet Of Things. El objetivo era promover el uso de protocolos de Internet en Objetos inteligentes. En otras palabras: hacer que el IoT se convirtiera en una realidad.

Ilustración 2: Logo de la IPSO Alliance

Actualmente, la IPSO Alliance continúa activa y en ella participan empresas como Google, Motorola, Bosch o Toshiba.

2.1.2 Arquitectura del Internet de las Cosas

Hoy en día no existe un único consenso sobre la arquitectura del Internet de las Cosas que sea aceptado universalemente, sino que los investigadores han propuesto diversos modelos de arquitectura:

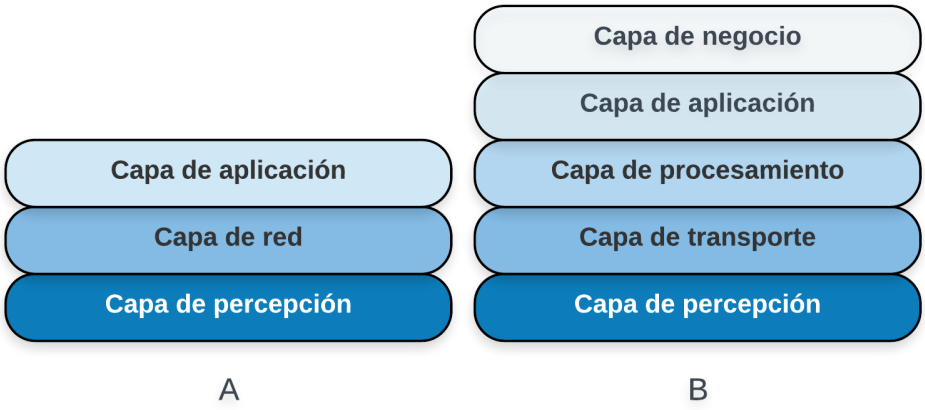

Ilustración 3: a) Arquitectura de tres capas del IoT. b) Aquitectura de cinco capas del IoT

a) Arquitectura de tres capas.

Se trata de la representación más sencilla, formada por las capas de percepción, de red y de aplicación.

La primera de ellas se corresponde con el nivel físico, englobando a los dispositivos y la adquisición de datos del entorno que éstos realizan por medio de sensores.

La capa intermedia se encarga de conectar estos dispositivos a otras "cosas inteligentes", a otros dispositivos, o a servidores de red y dispone de los mecanismos necesarios para transmitir datos entre ellos o realizar cierto grado de procesamiento de datos.

En la capa de aplicación se enmarcan las aplicaciones de usuario, donde los datos recogidos son utilizados con algún fin y pueden ser visualizados.

Esta arquitectura es perfecta desde el punto de vista conceptual, ya que describe al completo la lógica de la tecnología del IoT: los datos recabados por los dispositivos físicos se transmiten hacia otros dispositivos, servidores o elementos de la red y se procesan en las aplicaciones finales para un uso concreto. Sin embargo, a la hora de desarrollar aplicaciones reales, se precisa un mayor nivel de detalles, es decir, arquitecturas con más capas o conceptos diferentes.

b) Arquitectura de cinco capas.

En esta arquitectura hay dos capas, la de aplicación y la de percepción, que funcionan exactamente igual que en la arquitectura de tres capas. Adicionalmente, podemos encontrar las capas de transporte, de procesamiento y de negocio.

La primera de ellas se encarga de todo lo necesario para transmitir la información generada en la capa de percepción hasta la capa de procesamiento. Es decir, resuelve la comunicación entre dispositivos a nivel de red, utilizando una tecnología inalámbrica.

La capa de procesamiento recoge estos datos y, como parece obvio, los procesa independientemente de la tecnología anteriormente utilizada para obtenerlos. En esta capa se situarían todos los servicios de procesamiento de datos como bases de datos, cloud computing o big data, así que podemos decir que

es una de las capas principales de la arquitectura.

En la capa superior, la de negocio, s e resuelven todos los "problemas" de más alto nivel de abstracción, como los modelos de negocio, la privacidad de lso datos de usuario, y se gestionan las aplicaciones y en general todo el modelo IoT.

Existen otras arquitecturas diferentes, más complejas de entender, que quedan fuera del alcance de este proyecto. Conociendo los fundamentos, como son las arquitecturas de tres y cinco capas explicadas, podemos hacernos una idea sobre cómo funciona "por debajo" el Internet de las Cosas.

2.1.3 Retos y características del IoT

Los requisitos que deben cumplir los dispositivos y las redes de comunicación que implementen el IoT no están rigurosamente establecidos, si bien es cierto que en todos los escenarios deben cumplirse una serie de características esenciales. Son las siguientes:

- Dispositivos de bajo consumo energético. Los dispositivos conectados a la red deben ser diseñados para ser muy eficientes energéticamente, de forma que la autonomía de sus baterías les permita operar al menos varios años.
- Dispositivos de tamaño reducido. Los dispositivos deben tener las dimensiones adecuadas para ser utilizados en cualquier tipo de objeto.
- Movilidad. Los dispositivos y las redes involucradas deben diseñarse para permitir una total movilidad en las comunicaciones.
- Localización. Las redes diseñadas deben contar con mecanismos y sistemas para localizar a los dispositivos que forman parte de ellas. Se trata de un aspecto fundamental en muchos aplicaciones.
- Seguridad. Las comunicaciones deben asegurar la confidencialidad e integridad de los datos intercambiados. nuevos modelos de negocio, productos y compañías. Se espera que esta idea traiga consigo numerosos beneficios en aspectos como la optimización de la cadena de abastecimiento, efectividad de costos, mejoras en la experiencia de consumidores, beneficios en aspectos de seguridad y servicios de emergencia, etc.
- Amplio alcance de señal. Las tecnologías empleadas deben asegurar que la comunicación pueda realizarse a varios kilómetros y que acepten un elevado número de dispositivos. Además, deben ser tecnologías fáciles de implementar y administrar, sencillas y robustas.
- Dispositivos baratos. Las redes construídas en el IoT pueden llegar a estar formadas por miles de dispositivo, por lo que el coste de producción de los mismos, así como de los elementos de red debe ser bajo.

2.2 Tecnologías de comunicación en el IoT

Existen múltiples tecnologías de comunicación inalámbrica, pero no todas se adecúan a las características del Internet de las cosas. Las tecnologías inalámbricas del tipo WAN (GSM, UMTS, WIFI, etc.) no se ajustan a los requisitos anteriormente indicados por diversos motivos: el alcance de las conexiones, el coste de los dispositivos o la imposibilidad de prestar servicio a un gran número de participantes, son algunos de ellos.

Tampoco son adecuadas las tecnologías de redes locales de bajo consumo (como Bluetooth) ya que su alcance es demasiado limitado. Este problema sería subsanable mediante la creación de redes empleando topologías en malla; sin embargo, esta situación obligaría a la retransmisión constante de los datos, elevando el consumo de los dispositivos y agotando muy rápidamente sus baterías, de forma que se seguiría incumpliendo otro de los requisitos fundamentales del IoT.

Por este motivo, las únicas tecnologías aceptables para este propósito son aquellas que han sido desarrolladas con la aparición de las redes de sensores y el Internet de las Cosas. Se trata de las tecnologías Low-Power Wide Area Networks (LPWAN), que presentan una serie de características comunes:

- Todas operan en la banda libre ISM a una frecuencia inferior a 1GHz. Este tipo de señales presenta una menor atenuación que la banda de 2,4Ghz de ISM en presencia de obstáculos.
- Poseen esquemas de modulación que posibilitan la creación de enlaces de comunicación a varios kilómetros en zonas urbanas y decenas de kilómetros en zonas rurales, con un coste energético muy bajo. Principalmente se utilizan dos técnicas:
    - Modulación en banda estrecha y ultra estrecha. Consiste en codificar una señal en una banda lo más estrecha posible que permita su codificación. Este tipo de modulación tiene un nivel de ruido muy bajo y permite aumentar el alcance de la señal, a costa de disminuir la velocidad de transmisión. Además, la decodificación de la señal en el receptor es sencilla.
    - Modulación Spread Spectrum. Consiste en ensanchar la señal de forma que el ancho de banda de la señal que se transmite sea mucho mayor que el que se necesitaría para transmitir la señal original. De esta forma se consigue una transmisión más robusta frente a interferencias, aunque la decodificación en el receptor se hace más compleja y, por lo tanto, más costosa.

A continuación, se describen brevemente algunas de las redes de comunicaciones que han ido evolucionando orientadas hacia el sector del internet de las cosas.

2.2.1 Sigfox

Sigfox [2] es la red de comunicaciones LPWAN orientada específicamente al Internet de las Cosas más extendida a nivel mundial, cubriendo cerca del 98% del territorio Americano y Europeo. Utiliza una modulación de banda ultra estrecha y opera en la banda de 868MHz en Europa, y en la banda de 902MHz en Estados Unidos.

Uno de los motivos a los que debe su éxito hoy en día, además de su despliegue y cobertura casi global, es que los fabricantes de dispositivos IoT se han adaptado a esta tecnología, y además facilitan la subida de los datos recogidos a la nube de Sigfox, permaneciendo disponibles en los servidores de la compañía, lo que facilita su acceso a través de cualquier conexión a Internet. Este hecho, junto con el soporte que ofrece Azure de Microsoft hace que el proceso de desarrollo que agilice en gran medida.

Sigfox sigue un modelo de negocio totalmente propietario en el que ofrece todos los servicios del IoT: venta de dispositivos, red de conexión y servicios.

Ilustración 4: Logo Sigfox

2.2.2 LoRa

LoRa es otra red LPWAN que sigue un modelo de negocio similar al de Sigfox, aunque con una tecnología algo diferente, debido a que emplea un espectro de comunicaciones más amplio.

La diferencia más notable entre ambas redes es que LoRa está mucho mejor preparada para establecer una comunicación bidireccional en tiempo real entre dispositivos. Las especificaciones para los fabricantes que quieran desarrollar sus aplicaciones con LoRa también son bastante más relajadas y abiertas que las de Sigfox.

Por el contrario, la cobertura que ofrece LoRa es mucho menor; actualmente solo se encuentra desplegada en algunos países europeos como Francia, Bélgica, Suiza o los Países Bajos, además de en Sudáfrica. Sin duda, este es un factor bastante determinante a la hora de comenzar a desplegar un proyecto IoT.

LoRa es de código abierto y su empresa propietaria, Semtech, solo posee la capa física; la capa de acceso al medio se desarrolla de forma abierta gracias a la entidad LoRa Alliance. Es uno de los motivos por los que la tecnología LoRa representa una solución más atractiva para la realización de este proyecto que el resto de tecnologías.

Ilustración 5: Logo LoRa

2.2.3 Narrowband IoT

La Narrowband IoT [3] (NB IoT) es la tecnología LWPAN por la que apuestan las operadoras de telecomunicaciones a nivel global. Esto se debe a que su espectro de operación está dentro del rango del espectro del LTE o 4G, por lo que su despligue y explotación comercial está prácticamente asegurada aprovechado esta red.

Sin embargo, su despliegue, puesta en marcha y el estudio de las bondades de esta tecnología están menos desarrollados que el resto.

Ilustración 6: Logo Narrowband IoT

2.2.4 BLE

El BLE [4] o Bluetooth de baja energía (también conocido como Bluetooth Smart o Bluetooth Ultra Low Power) es otra tecnología inalámbrica empleada en determinados sectores de aplicación dentro del Internet de las Cosas. Permite la interoperación entre pequeños dispositivos cuando los datos a transmitir son reducidos, al contrario que el resto de tecnologías, capacitadas para enviar grandes volúmenes de datos.

Por este motivo, esta tecología se está utilizando principalmente en dispositivos pequeños que usan como batería pilas de botón, para dar servicios de localización o señalización y cuya vida puede durar meses debido a su baja tasa de transmisión de datos. Es el caso, por ejemplo, de los dispositivos beacons o balizas de localización. También tiene cabida en el escenario de la electrónica de consumo, como por ejemplo en los electrodomésticos de los hogares o los dipositivos wearables.

Ilustración 7: Logo Bluetooth Smart

2.2.5 Zigbee

Zigbee [5] no es una red de comunicaciones, sino una tecnología inalámbrica, bastante utilizada y centradas en aplicaciones industriales y domóticas. El motivo de aplicación de esta tecnología en este sector tan específico es su bajo consumo, la seguridad que ofrece (superior al resto de tecnologías), su robustez y su fácil escalabilidad, ofreciendo una gran capacidad para soportar un elevando número de nodos.

Además, la tasa de envío de datos en esta tecnología es baja, pero con un alcance de unos 100m, por lo que su uso en otro tipo de aplicaciones quedaría descartado, ya que surgiría la necesidad de utilizar numerosos enrutadores si los dispositivos se encontraran muy alejados del concentrador de datos.

Ilustración 8: Logo Zigbee

3 LA TECNOLOGÍA LORA

La tecnología LoRa ha logrado posicionarse como una de las principales tecnologías dentro del grupo de las LWPAN (Low Power Wide Area Networks) gracias a algunas de sus características, como son el servicio de comunicación bidireccional, la capacidad de interoperabilidad entre sensores distintos y la facilidad que ofrece a los desarrolladores y usuarios en el despliegue de aplicaciones IoT.

Cuando hablamos de LoRa en realidad debemos hacer referencia a dos tecnologías diferentes: LoRa y LoRaWAN. La primera de ellas define la capa física desarrollada por Cyleo en 2010, empresa que más tarde fue adquirida por Semtech; LoRaWAN es la especificación de red propuesta por la LoRa Alliance en 2015, que define una capa MAC basada en la modulación LoRa.

3.1 LoRa

LoRa es la capa física de la red LPWAN conocida como LoRaWAN. Como ya se ha indicado, la empresa Semtech [6] es la propietaria de LoRa, mientras que la capa de acceso al medio (estrictamente LoRaWAN) se desarrolla de forma abierta por la entidad sin ánimo de lucro LoRa Alliance [7].

En las siguientes secciones se va a describir la arquitectura y topología de esta tecnología, puesto que su comprensión es fundamental para entender y asimilar muchos de los aspectos de este proyecto.

3.1.1 Modulación en LoRa

LoRa opera en la banda ISM a una frecuencia inferior a 1GHz. Se trata de una banda libre que puede utilizarse sin licencia. En Europa se emplean las frecuencias de 868MHz y 433MHz.

El esquema de modulación empleado por LoRa es una variación de la modulación DSSS de tipo Spread Spectrum llamada Chirp Spread Spectrum. (CSS). Esta modulación permite crear conexiones de bajo coste y bajo consumo (unos 25mA en transimisión y 10mA en recepción) y robustas frente a interferencias. Estas modulaciones consisten en transmitir un mayor ancho de banda del realmente necesario, consiguiendo mayor robustez, pero a costa de hacer la decodificación de la señal más compleja.

LoRa permite realizar un ajuste de la tasa de transferencia y la potencia de transmisión de forma dinámica. Este mecanismo se denomina Adaptative Data Rate (ADR) y permite a los dispositivos ajustar sus parámetros en función del tamaño del mensaje a transmitir y la distancia a la pasarela, consiguiendo una mayor eficiencia y una velocidad de transmisión óptima.

Además, LoRa permite escoger entre seis factores de ensanchamiento en su modulación (SF o Spreading Factor). Cada uno de ellos define una relación entre potencia y tasa de transferencia. Mientras mayor sea el valor del SF mayor será la sensibilidad en el receptor y la distancia del enlace. Por contra, a mayor SF se obtendrán menores velocidades de transmisión.

3.1.2 Canales y rangos de frecuencia

LoRa puede trabajar en diferentes rangos de frecuencia, establecidas para distintas regiones del mundo:

- La banda que se emplea en Europa es la banda ISM de 863-870MHz regulada por la organización de estandarización europea ETSI. Utiliza un total de 8 canales elegidos aleatoriamente, con un ancho de banda de 0,3MHz cada uno.
- En Estados Unidos, Canadá, Australia, Singapur o Israel se emplea la banda ISM de 902 – 928 MHz, utilizando 13 canales con un ancho de banda de 2,16MH.

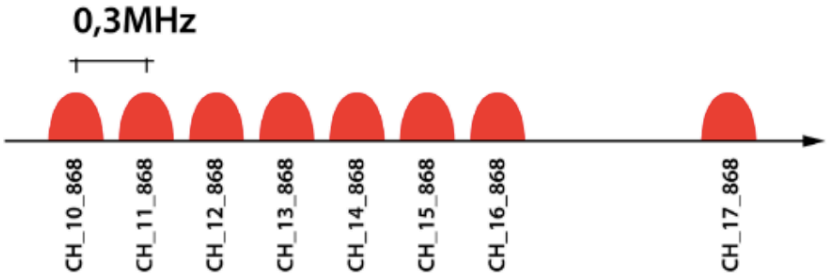

Ilustración 9: Banda de frecuencias LoRa en Europa

3.2 LoRaWAN

Como ya se ha indicado, LoRaWAN define la capa de acceso al medio, el protocolo de comunicación y la arquitectura de red de la tecnología LoRa. Se trata de una capa de acceso al medio compleja, debido a las posibilidades y funcionalidades que ofrece: soporte de numerosos tipos de dispositivo, activación y configuración de los dispositivos en la red e implementación de medidas de seguridad que aportan integridad y confidencialidad en el intercambio de mensajes.

Tanto el protocolo de comunicación como la arquitectura de red son los factores que mayormente determinan la duración de la batería de un nodo, la capacidad de la red, la calidad del servicio, la seguridad, y la variedad de aplicaciones que pueda servir la red.

3.2.1 Arquitectura de una red LoRaWAN

La red LoRaWAN está formada por tres elementos:

- Dispositivos finales o clientes. Se trata de aquellos dispositivos empleados para la conexión de los diversos objetos a la red LoRa, recogiendo información y enviándola a la pasarela de la red.
- Pasarelas o concentradores. Se trata de estaciones base LoRa que reciben la información obtenida por los múltiples dispositivos finales y las reenvían a los servidores de red.
- Servidores de red. Se trata de equipos encargados de la recepción y el procesamiento de la información reenviada por las pasarelas. También se encargan de la gestión y configuración de la red y los dispositivos finales.

Normalmente, el conjunto de dispositivos finales y la pasarela se agrupan formando una conexión con topología en estrella. Las transmisiones en este tipo de topología no necesitan reenvíos y son fáciles de implementar y gestionar. Por otra parte, las pasarelas se conectan a los servidores actuando en forma de puentes, resultando igualmente un diseño de red bastante sencillo.

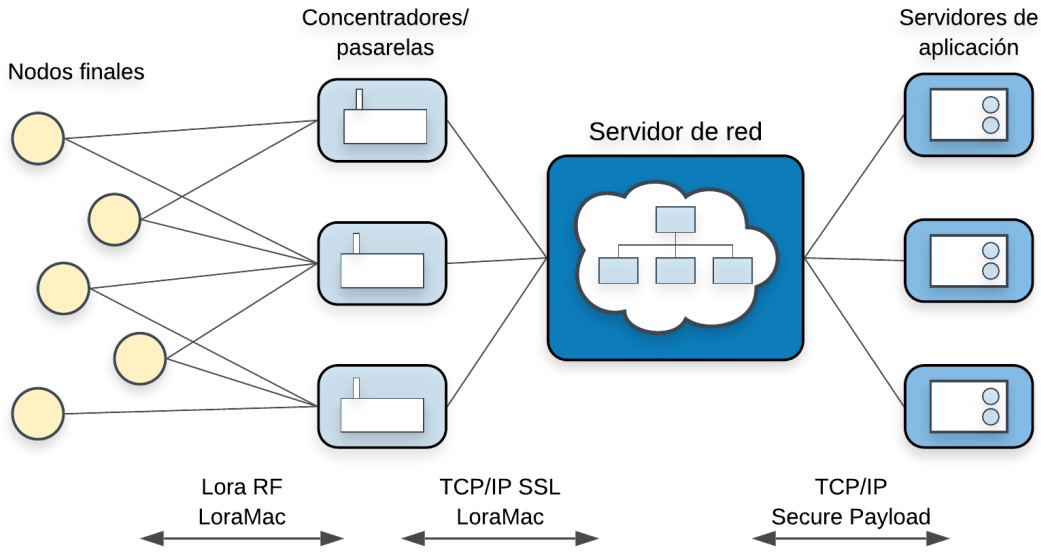

Ilustración 10: Arquitectura de una red LoRaWan

En otros tipos de redes inalámbricas, los nodos finales reenvían la información de otros nodos para incrementar el rango de comunicación y el tamaño de la célula de la red. A pesar de que el alcance aumenta, también se añade mucha complejidad, se reduce la capacidad de la red y la vida de las baterías.

LoRaWan evita esta complejidad utilizando una arquitectura de red en estrella, logrando así optimizar los recursos de los dispositivos y un mayor alcance en las comunicaciones.

Los nodos no van a estar necesariamente asociados a una pasarela, sino que sus datos podrán ser recibidos por varias pasarelas a la vez. Cada una de ellas enviará la información recibida al servidor de la red mediante alguna otra tecnología (ya sea WiFi, red móvil, ethernet o comunicación vía satélite).

La inteligencia de la red reside en dicho servidor, encargado de gestionar la red y eliminar la información redundante filtrando los paquetes recibidos, realizando chequeos de seguridad, realizando adaptaciones en la tasa de datos, etc.

Por otra parte, la comunicación entre los nodos es asíncrona; los nodos envían la información cuando la tienen disponible. Este método se conoce también como Aloha. En una red con topología en malla o con comunicación síncrona, los dispositivos tienen que “despertar” y sincronizarse con la red antes de comprobar si pueden recibir mensajes. Esta sincronización consume mucha energía y es la mayor causa de pérdida de batería de los dispositivos.

3.2.2 Capacidad de una red LoRaWAN

La pasarela de la red debe poseer una gran capacidad de recibir mensajes desde un elevado número de nodos, a fin de lograr el mayor tamaño posible de la red en estrella, obteniendo así un mayor rango de comunicación. En LoRaWAN esta capacidad se consigue empleando tasas de datos adaptativas y usando transceptores multicanal en la pasarela, de manera que pueda recibir y procesar múltiples paquetes

simultáneamente procedentes de distintos canales. Por tanto, los factores críticos a la hora de determinar la capacidad de la red son:

- El número de canales concurrentes empleados.
- La longitud del payload o carga útil del paquete.
- La frecuencia de transmisión de los distintos nodos.
- La tasa de datos. Esta tasa varía según el "spreading factor" de la modulación empleada. Debido a que LoRa usa modulación de espectro ensanchado, cuando se utilizan distintos factores de ensanchado para las señales, éstas resultan ser prácticamente ortogonales entre sí. Es por ello que cuando este factor cambia, también lo hace la tasa de datos. La pasarela es capaz de recibir múltiples señales en un mismo canal con distintas tasas de datos simultáneamente.

3.2.3 Canales y velocidades de transmisión

La comunicación entre los nodos de la red y la pasarela, se realiza empleando distintos canales de frecuencia y tasas de datos. Se ha explicado que, haciendo uso de la modulación de espectro ensanchado, las comunicaciones con distintas velocidades de datos no interfieren entre sí, creando un conjunto de canales virtuales que aumentan la capacidad de la red.

Por tanto, los nodos pueden transmitir por cualquier canal disponible, en cualquier momento, y empleando cualquier tasa de velocidad que sea posible siempre y cuando se cumplan una serie de características:

- El nodo transmisor cambiará de canal de forma aleatoria para cada transmisión. Esto hace que el canal sea mucho más robusto frente a interferencias.
- El nodo escogerá el máximo ciclo de trabajo de transmisión según la sub-banda empleada y respetando las regulaciones locales. Sucede lo mismo para la duración máxima de transmisión.

Para las bandas de 433MHz y 868MHz (es el caso de Europa) se emplean 16 canales. En el caso de la banda de 900MHz se encuentran disponibles 72 canales.

En la tabla 1 se recogen la información sobre los parámetros para los canales según la norma aplicada en Europa.

Los primeros canales tienen frecuencias asignadas, mientras que el resto son configurables dentro del rango asignado.

Canal	**Parámetros**	**Bandas de frecuencia**	
		868 MHz	**433 MHz**
0	Frecuencia	868100 kHz	433175 kHz
	Ciclo de trabajo	0.33%	0.33%
	Tasa de datos	0 - 5	0 - 5
	Estado	On	On
1	Frecuencia	868300 kHz	433375 kHz
	Ciclo de trabajo	0.33%	0.33%
	Tasa de datos	0 - 5	0 - 5
	Estado	On	On
2	Frecuencia	868500 kHz	433575 kHz
	Ciclo de trabajo	0.33%	0.33%
	Tasa de datos	0 - 5	0 - 5
	Estado	On	On

3-15	Frecuencia	868325 kHz - 869750 kHz	433050 kHz - 434790 kHz
	Ciclo de trabajo	*	*
	Tasa de datos	0 - 5	0 - 7
	Estado	Off	Off

Tabla 1: Canales y parámetros LoRaWAN

Los canales del 3 al 15 por defecto están en estado Off. Al activarlos, se deben configurar adaptando el ciclo de trabajo del resto de canales, de manera que el total de canales no sobrepase el 1% máximo permitido en la banda de frecuencias. Además, estos canales son los únicos que pueden ser configurados con las tasas de datos 6 y 7, a diferencia de los canales configurados por defecto.

*Hay limitaciones que establecen cuánto tiempo puede estar el transmisor activado o el tiempo máximo que puede estar transmitiendo. LoRaWAN impone una limitación del ciclo de trabajo por cada sub-banda. Cada vez que una trama se transmite en una sub-banda dada, el tiempo de la emisión y la duración en el aire de la trama se registran para esta sub-banda. La misma sub-banda no puede ser utilizada de nuevo durante un intervalo de timpo, que vendrá determinado por el tiempo en el aire y el ciclo de trabajo de la sub-banda.

La tasa de datos de la tabla anterior viene designada por los parámetros recogidos en la tabla 2:

Tasa de datos	Configuración	Tasa de bit
0	LoRa: SF12 / 125 kHz	250 bps
1	LoRa: SF11 / 125 kHz	440 bps
2	LoRa: SF10 / 125 kHz	980 bps
3	LoRa: SF9 / 125 kHz	1760 bps
4	LoRa: SF8 / 125 kHz	3125 bps
5	LoRa: SF7 / 125 kHz	5470 bps
6	LoRa: SF7 / 250 kHz	11000 bps
7	FSK: 50 kbps	50000 bps

Tabla 2: Configuración y tasas de datos LoRaWAN

El SF o Spreading Factor (factor de ensanchamiento) determina la cantidad de datos redundantes que se envían en la transmisión. La velocidad viene condicionada por este factor ya que, mientras mayor sea éste, mayor cantidad de datos redundantes se enviarán, mayor será la robustez de la comunicación, y mayor el alcance de la misma, a costa de una menor velocidad de transmisión.

3.2.4 Clases de dispositivos LoRaWAN

En una red LoRaWAN los dispositivos se dividen en tres clases según las funcionalidades que soportan. Las tres clases pueden coexistir en la misma red y los dispositivos pueden cambiar de clase con tan solo cambiar su configuración.

- **Clase A**. Se trata de dispositivos bidireccionales ue permiten la transmisión bidireccional de datos. Por ello, después de cada trasnmisión ascendente por parte del dispositivo, siempre hay dos ventanas de recepción para recibir una respuesta descendente. El propio dispositivo final planifica las transmisiones y la recepción de mensajes solo está permitida tras una transmisión finalizada exitosamente, por lo que si ésta no se produce, para recibir algún dato se debería esperar hasta la siguiente transmisión ascendente. Se trata de la clase más eficiente desde el punto de vista energético y es la más adecuada cuando la aplicación solo requiere enviar datos.
- **Clase B**. En esta clase se pueden crear ventanas de recepción adicionales sin necesidad de que anteriormente haya existido una transmisión, aumentando así la capacidad de recibir datos por parte del dispositivo. La pasarela puede sincronizarse con el dispositivo final a través del envío de tramas beacon y, de esta forma, planificar el tiempo que éste debe mantener abierta la ventana de recepción. Este modo de funcionamiento implica un mayor consumo energético.
- **Clase C**. Los dispositivos se mantienen constantemente en estado de recepción, el cual solo se interrumpe si se produce una transmisión. Este modo de funcionamiento implica un gran consumo de energía.

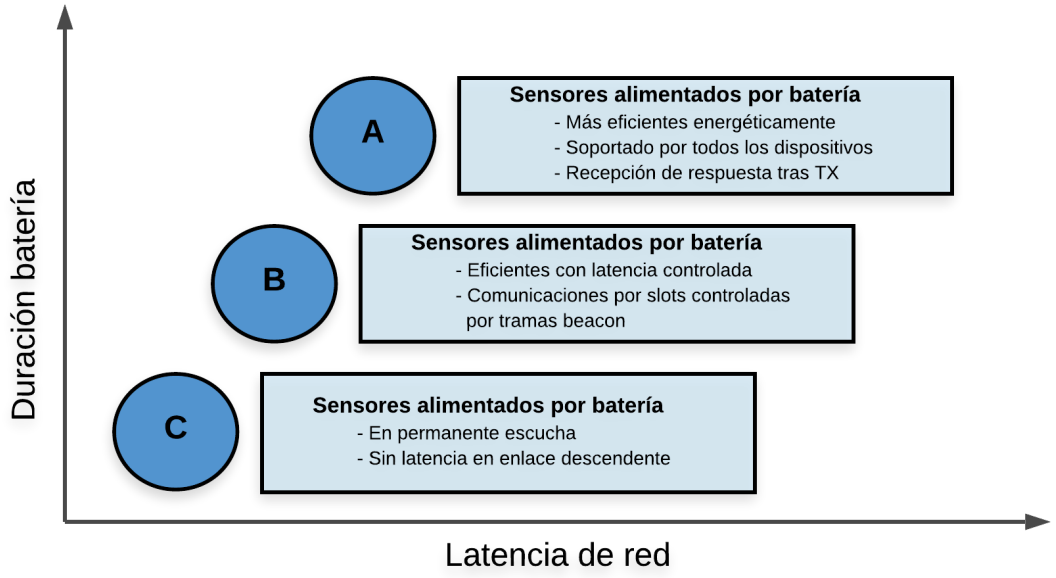

Ilustración 11: Clases de dispositivo LoRaWAN

Como puede apreciarse en la figura 11, las clases representan un balance entre consumo energético y capacidad de recepción, y deberán escogerse adecuadamente en función de la aplicación a desarrollar según la demanda de datos necesaria.

3.2.5 Seguridad en LoRaWAN

Es de extrema importancia en muchas aplicaciones incorporar seguridad a la red LPWAN. LoRaWAN emplea dos capas de seguridad: una para la red y otra para la aplicación. Se emplea para ello un cifrado AES con intercambio de claves, usando un identificador basado en la norma IEEE EUI64.

- Capa de seguridad de red: se logra utilizando una clave de 128 bits que garantiza la seguridad a nivel de red y la autenticidad de los nodos en ella.
- Capa de seguridad de aplicación: garantiza que el operador de red no tenga acceso a los datos de usuario de la aplicación. Se emplea una clave de 128 bits que garantiza la seguridad extremo a extremo a nivel de aplicación.

3.2.6 Estructura de un paquete LoRaWAN

Todos los paquetes LoRaWAN están formados por un preámbulo de 8 bytes de longitud, una cabecera física (PHDR) y el payload. Tanto la cabecera como el payload tienen un código de corrección de errores o CRC.

PREÁMBULO	PHDR	PHDR-CRC	PAYLOAD	CRC

Tabla 3: Estructura de una trama LoRaWAN

El Payload de la capa física cuenta con una cabecera MAC (MHDR), el MAC Payload y un Message Integrity Code (MIC). Se trata de un código de cuatro bytes que se calcula a partir de la Network Session Key y es una clave para autentificar el mensaje generada a partir de los campos MHDR y MACPayload.

MHDR	MACPayload	MIC
1 byte	1-M bytes	4 bytes

Tabla 4: Estructura del Payload de una trama física LoRaWAN

La cabecera MAC o MHDR especifica el tipo de mensaje y la versión del formato de la trama de la especificación de la capa LoRaWAN con la que ha sido codificada. Existen 6 tipos de mensajes MAC, recogidos en la tabla 5.

Tipo	**Descripción**
000	Join Request
001	Join Accept
010	Unconfirmed Data Up
011	Unfoconfirmed Data Down
100	Confirmed Data Up
101	Confirmed Data Down
110	RFU
111	Propietary

Tabla 5: Tipos de mensaje MAC en LoRaWAN

El MAC Payload o MHDR está formado por una cabecera de trama (FHDR), un campo de puerto opcional y un campo de Payload de trama opcional, tal y como se muestra en la tabla 6.

FHDR	FPort	FRM Payload
7-23 bytes	0-1 bytes	0-N bytes

Tabla 6: Estructura del MacPayload de una trama LoRaWAN

La cabecera de la trama o FHDR contiene la dirección con la que se identifica el dispositivo dentro de la red LoRaWAN, el campo FCtrl para habilitar la tasa adaptativa de datos, un contador de tramas y un campo FOpts en el caso de que lo que se transmita sea un comando MAC. El campo FPort sirve para especificar si el campo FRM Payload contiene comandos MAC o datos de aplicación. Esta estructura se muestra en la tabla 7.

DevAddr	FCtrl	FCnt	FOpts
4 bytes	1 byte	2 bytes	0 – 15 bytes

Tabla 7: Estructura de la cabecera de trama FHDR en LoRaWAN

Los 7 bits más significativos del campo DevAddr sirven para identificar la red (NwkID), mientras que los 25 restantes se corresponden con la dirección de red (NwkAddr), la cual puede ser configurada y escogida por el administrador de red.

Teniendo en cuenta todo esto, el tamaño máximo del MacPayload (M en la tabla) puede variar según la banda de frecuencia sobre la que se trabaja, la tasa de datos, o la ausencia del campo de control FOpts (de tamaño N), tal y como se muestra en la tabla 8 (datos para la banda de 868MHz utilizada en Europa).

Tasa de datos	**M (bytes)**	**N (bytes)**
0	59	51
1	59	51
2	59	51
3	123	115
4	230	222
5	230	222
6	230	222
7	230	222

Tabla 8: Tamaño máximo del MacPayload para distintas tasas de datos y tamaños de FOpts

4 LA PLATAFORMA FIWARE

Fiware [8] es una infraestructura en la nube desarrollada para facilitar la creación y el despliegue global de servicios y aplicaciones inteligentes, promovida por la industria TIC europea y la Comisión Europea.

Esta plataforma forma parte de los proyectos de la FI-PPP (Future Internet Public Private Partnership) que está constituida, además, por otros proyectos dentro de esta iniciativa, que comparten el objetivo de lograr un Internet más accesible y útil para los ciudadanos, el "internet del futuro".

Para lograr su objetivo, FIWARE proporciona una serie de APIS simples y potentes con las que desarrollar de manera sencilla aplicaciones inteligentes en múltiples sectores. Dichas APIs y herramientas son de código abierto.

4.1 Introducción a Fiware

Fiware es una plataforma de código abierto que combina componentes que permiten la conexión del Internet de las Cosas con los servicios de administración de información de context y Big Data en la nube.

Ilustración 12: Logo Fiware

Fiware ofrece APIs estándar para la gestión y el intercambio de datos, así como modelos de datos reutilizables en cualquier proyecto o aplicación inteligente.

Además, permite realizar la automatización de procesos durante todo el ciclo de vida de la aplicación, permitiendo la interoperabilidad y facilitando la integración de la misma con otras soluciones y servicios de terceros.

La plataforma Fiware también hace de punto de encuentro para desarrolladores, empresarios, inversores, etc. a través de la Comunidad Fiware, que se ha extendido ya a más de 100 ciudades y tiene varias asociaciones estratégicas como GSMA, TM Forum, CEF y ETSI, entre otras. Además, cuenta con 11 iHubs y diversos programas de aceleración.

Es por ello que la comunidad FIWARE no solo se compone de la plataforma FIWARE en sí, sino que también está formada por un conjunto de organizaciones, las cuales son:

- **Plataforma FIWARE:** proporciona una serie de APIS simples y a la vez potentes y de código abierto, que facilitan el desarrollo de aplicaciones inteligentes en muchos sectores. En la plataforma pueden encontrarse además implementaciones de referencias para cada uno de los componentes FIWARE, de forma que el desarrollador pueda implementar sus soluciones más rápida y cómodamente.
- **FIWARE Lab**: se trata de un entorno no comercial dedicado a la innovación y la experimentación basada en la tecnología FIWARE. Está desplegado a lo largo de una red geográficamente distribuida de nodos federados sobre un amplio rango de infraestructuras experimentales.
- **FIWARE Accelerate**: se trata de un programa de aceleración que promueve la adopción de la tecnología FIWARE entre los integradores y desarrolladores de soluciones y aplicaciones, centrándose principalmente en pequeñas y medianas empresas y start-ups. Vinculado a este programa, en 2014 la UE lanzó una campaña muy ambiciosa para movilizar 80 millones de euros para apoyar a las pymes y empresarios que desarrollarán aplicaciones innovadoras basadas en FIWARE.
- **FIWARE Mundus**: a pesar de surgir en Europa, FIWARE fue diseñada con ambición global, por lo que los beneficios de su uso pueden ser aprovechados en otras regiones. Este programa está diseñado para que FIWARE llegue a distintas partes del mundo, como Norte América, Latinoamérica, África o Asia.
- **FIWARE iHubs:** esta red tiene como objetivo formar una red de desarrolladores, actuando de forma local. Actualmente existen 11 nodos iHubs repartidos por el mundo centrados en la creación y las operaciones Fiware.

El objetivo principal de FIWARE es, por tanto, ayudar al desarrollo e implementación de nuevos servicios, proporcionando un conjunto de APIs para el desarrollo rápido de aplicaciones en numerosos sectores, facilitando la reutilización e introduciendo estándares. Separa la cadena de valor de las aplicaciones, de forma que la plataforma, el desarrollo de los servicios, su despliegue, etc. puedan ser proporcionados por entidades diferentes, ayudando de esta forma a su expansión y mejorando la competitividad para acelerar la expansión comercial.

4.2 El catálogo Fiware

FIWARE está basado en un conjunto de elementos denominados “Enablers genéricos” (Generic Enablers – GE). Se trata esencialmente de una serie de programas reutilizables que proporcionan diversas funcionales en función de las características del proyecto a realizar.

Estos elementos se organizan dentro del **catálogo FIWARE,** donde puede encontrarse toda la librería de enablers junto a implementaciones de referencia, que permiten a los desarrolladores poner en marcha diversas funcionalidades, haciendo la programación mucho más sencilla.

El principal componente y único obligatorio en cualquier plataforma “impulsada por FIWARE” es el enabler **Orion Context Broker**, el cual realiza una función angular en cualquier solución inteligente: gestiona la información de contexto, permitiendo la realización de actualizaciones y el acceso al contexto.

Construídos alrededor de este Context Broker se encuentran una gran variedad de componentes Fiware complementarios, que se encargan de diversos aspectos, como los siguientes:

- Interfaces con el Internet de las Cosas o IoT, Robots y sistemas de terceros, que permiten capturar actualizaciones de la información de contexto y traducir cualquier acción requerida.
- Captura de datos de contexto/administración de APIs, publicación y monitorización, implementando los comportamientos esperados en aplicaciones inteligentes y/o asistiendo a los usuarios finales para tomar decisiones inteligentes.
- Procesado, análisis del tipo “Big Data” y visualización de información de contexto, brindando soporte para el control de uso y la oportunidad de publicar y monetizar parte de los datos de contexto que son administrados.

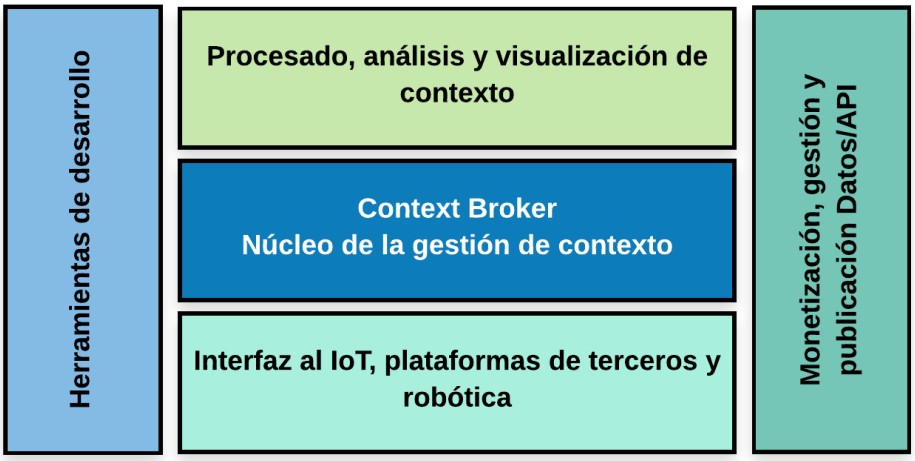

Ilustración 13: Arquitectura Fiware

Estos enablers genéricos complementarios no tienen por qué usarse, sino que junto al Context Broker pueden diseñarse plataformas híbridas que trabajen con componentes de terceros, a elección del desarrollador. Simplemente basta con usar la tecnología que proporciona el Context Broker para gestionar la información de contexto para que la aplicación desarrollada pueda catalogarse como "impulsada por FIWARE".

En capítulos posteriores de este documento se detallará la implementación del Context Broker así como de otro enabler que proporcionará funcionalidad añadida a la aplicación que se va a desarrollar.

4.3 Principios básicos de diseño

A la hora de diseñar una aplicación basada en Fiware y que utilice el Context Broker, es necesario tener en cuenta los siguientes prinpicios de diseño:

- **Persistencia del contexto**. La información recibida en el Context Broker (desde una fuente de contexto o como resultado de una petición a un Proveedor de contexto) se almacena en una base de datos de contexto. Si algún otro consumidor necesita acceder a esa misma información, ya se encontrará almacenada en la base de datos. La persistencia en la base de datos solo se mantiene para el estado de la información en el un determinado instante, es decir, el Context Broker no almacena un histórico del estado de la información de contexto. Sin embargo, Fiware aporta otras herramientas para lograr este propósito.
- **Escalabilidad**. La información almacenada puede llegar a ser bastante voluminosa, por lo que el Context Broker debe estar diseñado teniendo en cuenta la escalabilidad, así como la tecnología usada por la base de datos.
- **Geolocalización**. Una entidad puede estar asociada a una determinada ubicación (generalmente coordenadas GPS), por lo que el Context Broker debe estar preparado para manejar este tipo de información.
- **Federación**. Un Context Broker puede tener el papel de consumidor o proveedor en otro escenario, de manera que puedan formarse arquitecturas federadas.
- **Paginación**. El resultado de una petición síncrona por parte de un consumidor de context puede ser muy pesado, lo que hace muy complicado incluir el resultado en una sola respuesta. Por ello el Context Broker incluye mecanismos de paginación mediante los cuales devuelve la respuesta en bloques de información.
- **Multitenancy**. El Context Broker debe implementar mecanismos para aislar los entornos de

consumidor y proveedor, de forma que el conjunto de consumidores/proveedores no pueda acceder al contexto al que pertenecen el resto de consumidores/proveedores.

- **Seguridad**. Esta característica no es obligatoria de forma nativa, pero puede implementarse otros enablers de Fiware.

4.4 Orion Context Broker

4.4.1 Definiciones y conceptos

Para comprender las interacciones que se producen en la arquitectura del Context Broker, es necesario conocer algunos términos y conceptos básicos relacionados con el mismo.

Dentro del modelo de publicación/suscripción que sigue el Context Broker exiten una serie de actores básicos:

- **Información o elemento de contexto:** se trata de una estructura de datos genérica que hace referencia a toda aquella información que se produce, recopila u observa, que puede ser relevante para ser procesada, para ser analizada o para obtener algún tipo de conocimiento. Cada intercambio de contexto en el escenario en el que nos encontramos hace referencia a una determinada entidad, que a su vez puede formar parte de un grupo de entidades más complejo.
- **Context Broker (CB)**: se trata del componente principal de la arquitectura. Se encarga de manejar y agregar los datos de contexto y actúa como interfaz entre los distintos actores de la arquitectura. Principalmente debe encartgarse de controlar el flujo de datos entre actores, para lo cual debe conocer a los diferentes Proveedores de Contexto del escenario. Para realizar esta tarea, el CB proporciona servicios de búsqueda de proveedores y de persistencia de datos de contexto.

- **Productor de contexto (CP):** se trata del componente capaz de generar o actualizar contexto. Para ello se produce una comunicación entre el CP y el CB.

- **Proveedor de contexto (CPr):** se trata de un productor de contexto específico que proporciona información bajo demanda de forma síncrona, es decir, siendo invocado por el Context Broker o por el Consumidor.

- **Consumidor de contexto (CC):** se trata de la entidad que utiliza y explota los datos de contexto, que puede obtener bien realizando una petición al CB o bien invocando directamente al CP a través de su interfaz específica. También es posible recuperar la información de contexto a través de la suscripción a eventos cuando se cumplen ciertas condiciones; en este caso es el Context Broker el que notifica al consumidor cuando se generan actualizaciones relevantes.

4.4.2 Funcionalidad del Context Broker

Como ya se ha referido, el Context Broker permite manejar y gestionar la información de contexto de nuestra aplicación durante todo su ciclo de vida, incluidas actualizaciones, consultas y suscripciones a cambios en la información de contexto, y lo hace de forma altamente descentralizada y a gran escala.

El Context Broker permite publicar la información generada por las entidades productoras, de forma que esté disponible y fácilmente accesible por las entidades consumidoras.

Por otra parte, las entidades consumidoras solamente se encargarán de consumir información de su interés, sin necesidad de conocer qué entidad la generó, su localización o el protocolo nativo para recuperar dicha información; es decir, solamente estarán interesadas en el evento en sí, en lugar de su origen.

Por tanto, el principio fundamental del Context Broker es que se logra desacoplar totalmente las entidades productoras y las consumidoras, de forma que las primeras pueden publicar datos sin saber qué, dónde y cuándo serán utilizados por las entidades consumidoras. De esta forma, las entidades no tienen que estar necesariamente conectadas y, por ello, no necesitan utilizar los mismos protocolos, logrando la interoperabilidad.

Es importante aclarar que el Context Broker nos permite guardar información sobre el contexto actual únicamente. Es decir, este agente solo dispone de la última "foto" de los datos capturados.

Sin embargo, debido a que la información evoluciona constantemente a lo largo del tiempo, sería necesario crear un historial de la misma a fin de desarrollar la aplicación que se pretende. Para poder llevar a cabo una monitorización eficaz, se deben emplear otros enablers que complementen al Context broker y permitan la persistencia de la información.

4.4.3 La interfaz NGSI

Para comunicarse con el Context Broker, éste implementa la interfaz. Se trata de una interfaz basada en la especificación NGSI Open 2 Mobile Alliance (OMA), que permite manejar y manipular cualquier tipo de datos, y gestionar toda la información de contexto.

La Ilustración 14 muestra la arquitectura lógica de comunicación con el Context Broker, así como los principales componentes, interacciones y actores que intervienen en el escenario.

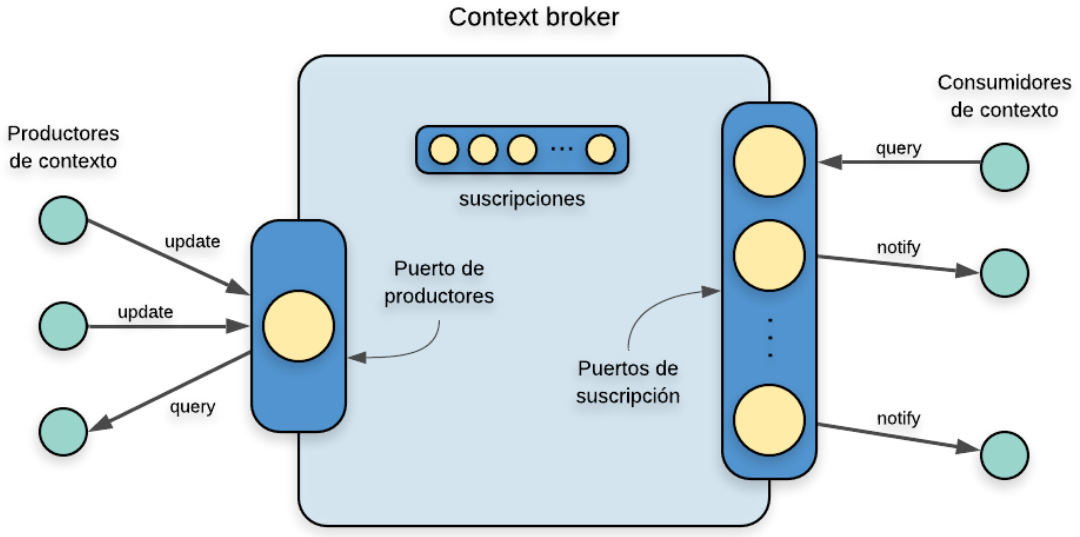

Ilustración 14: Arquitectura lógica del Context Broker

La interfaz NGSI (Next Generation Service Interface) define:

- Un modelo de datos para la información de contexto, basada en la definición de entidades de contexto. Cualquier desarrollador puede crear un modelo de datos que se ajuste a las necesidades de su proyecto siguiendo la norma especificada por NGSI, si bien es cierto que ya se encuentran definidos numerosos modelos que pueden ajustarse a las necesidades del mismo. Dichos modelos ya se encuetran armonizados de manera que se logre la interoperabilidad entre aplicaciones.
- Una interfaz de datos de contexto que permita el intercambio de información a través de operaciones de consulta, suscripción y actualización de los mismos.

4.4.3.1 Representación de una entidad NGSI

Las entidades en NGSI se representan mediante un objeto JSON (JavaScript Object Notation) con la siguiente sintaxis:

- Un identificador y tipo de entidad (*EntityId* y *EntityType*) que identifican de manera única a la entidad a la que hacen referencia los datos de contexto.
- Una secuencia de uno o más atributos identificados por las tuplas <*nombre, tipo, valor*>.
- Meta-datos opcionales unidos a los atributos, que siguen la misma estructura.

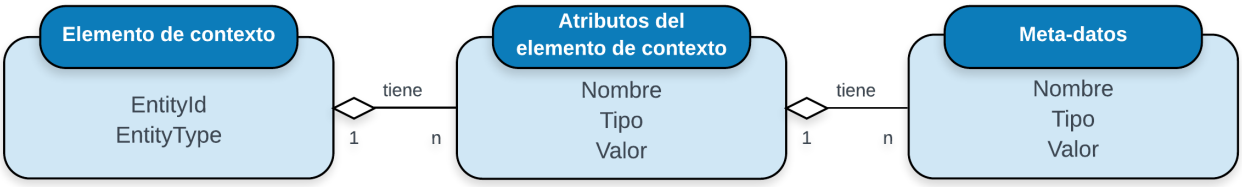

Ilustración 15: Estructura de una entidad

A continuación, se muestra un ejemplo genérico de la sintaxis de definición de una entidad.

```
{
"id": "entityID",
"type": "entityType",
"attr_1": <val_1>,
"attr_2": <val_2>,
...
"attr_N": <val_N>
}
```

Y la sintaxis para definir un elemento atributo:

```
{
"value": <...>,
"type": <...>,
"metadata": <...>
}
```

La representación normalizada de los atributos de una entidad siempre incluye el id, el tipo, y las propiedades que representan a los atributos. Sin embargo, existe una representación parcial más simple definida por duplas atributo-valor. En este caso los atributos de las entidades se representan únicamente por sus valores, excluyendo la información sobre el tipo y los metadatos. A continuación, se muestra un ejemplo genérico de este tipo de sintaxis:

```
{
  "id": "R12345",
  "type": "Room",
  "temperature": 22
}
```

4.4.3.2 Interacciones entre entidades en NGSI

En este apartado se describen las interacciones de comunicación entre los distintos actores dentro del escenario FIWARE. Mediante su uso, los productores y consumidores podrán:

- Registrar contexto producido por aplicaciones (por ejemplo, un sensor de temperatura ubicado en una habitación).
- Actualizar dicha información de contexto (por ejemplo, mandando actualizaciones de la temperatura).
- Recibir notificaciones cuando se produzcan cambios en la información de contexto (por ejemplo, cuando la temperatura cambia) o recibir notificaciones a una determinada frecuencia (por ejemplo, tomar la temperatura a cada minuto).
- Consultar la información de contexto almacenada. El Context Broker almacena la información actualizada procedente de las aplicaciones, por lo que las consultas se resolverán basándose en dicha información.

A continuación, se describen cada una de las interacciones. En todas ellas, como ya se ha visto, el intermediario entre productores y consumidores es el Context Broker.

a) Interacciones básicas y relación entre entidades

El esquema de la Ilustración 16 muestra el diagrama básico de comunicación entre entidades productoras y consumidoras.

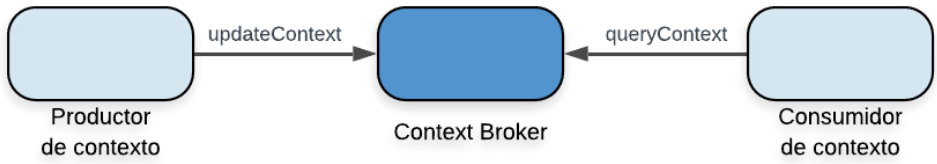

Ilustración 16: Interacciones básicas en el esquema publicación/suscripción del Context Broker

Los productores de contexto publican datos o elementos de contexto invocando la operación *updateContext* en el Context Broker.

Los consumidores pueden obtener dicha información a través de la invocación de la operación *queryContext* en el Context Broker.

Los datos de contexto persisten en el Context Broker y están preparados para ser consultados en cualquier momento. Esta es la característica que diferencia esta arquitectura de otros estándares.

b) Interacciones relativas a los productores de contexto

Como ya se ha indicado anteriormente, los Productores de contexto publican información invocando la operación *updateContext* en el Context Broker. Algunos productores (denominados Proveedores de contexto) pueden también exportar la operación *queryContext* como resultado del reenvío de una petición de contexto por parte de un consumidor.

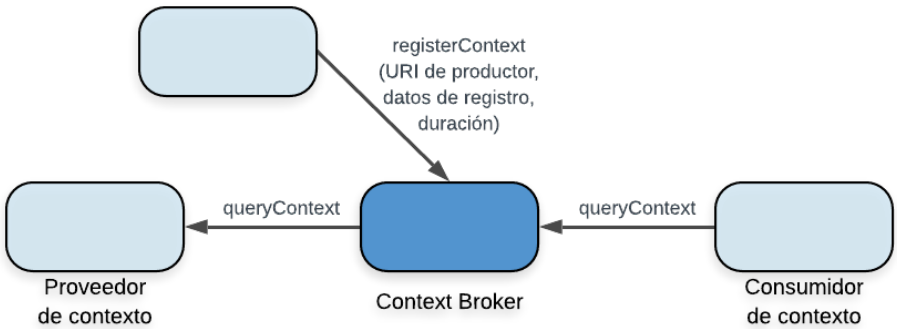

Ilustración 17: Interacciones relativas a los Productores de contexto

Los consumidores de contexto pueden obtener la información deseada invocando la operación *queryContext* en el Context Broker.

El Context Broker puede reenviar dicha petición a un Proveedor de contexto apropiado y devolver el resultado al Consumidor de contexto original.

c) Interacciones relativas a los consumidores de contexto

El Context Broker reenvía la información de contexto a los consumidores que la precisen.

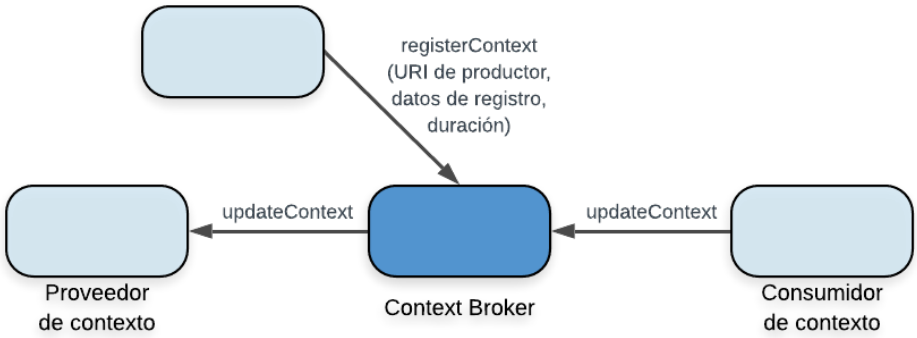

Ilustración 18: Interacciones relativas a los consumidores de contexto

Los productores de contexto pueden enviar una operación de *updateContext* al Context Broker.
El Context Broker reenvía el *updateContext* al consumidor apropiado y devuelve el resultado (éxito o error) al Proveedor de contexto original.

En determinadas ocasiones resulta útil recibir información cuando se cumplen ciertas condiciones. Para ello los consumidores pueden suscribirse a eventos usando la operación *subscribeContext* durante un determinado tiempo.

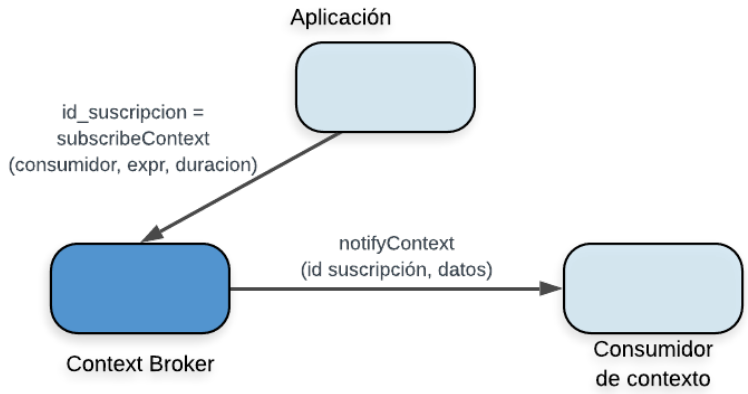

Ilustración 19: Interacciones relativas a la suscripción de eventos de los consumidores de contexto

Los consumidores suscritos a eventos reciben información espontáneamente a través de la operación *notifyContext*.

d) Operaciones avanzadas

La operación registerContext no solo se utiliza para registrar información de contexto en el Context Broker, sino también para registrar la existencia de entidades del sistema y la disponibilidad de los distintos atributos.

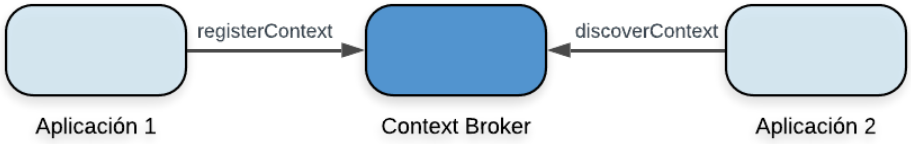

Ilustración 20: Interacciones relacionadas con el registro de entidades y atributos

Al igual que ocurría con la suscripción a eventos acerca de los datos de contexto, también es posible realizar suscripción al registro de entidades o a la disponibilidad de determinados atributos que cumplen con ciertas condiciones, a través de la operación *subscribeContextAvailability*.

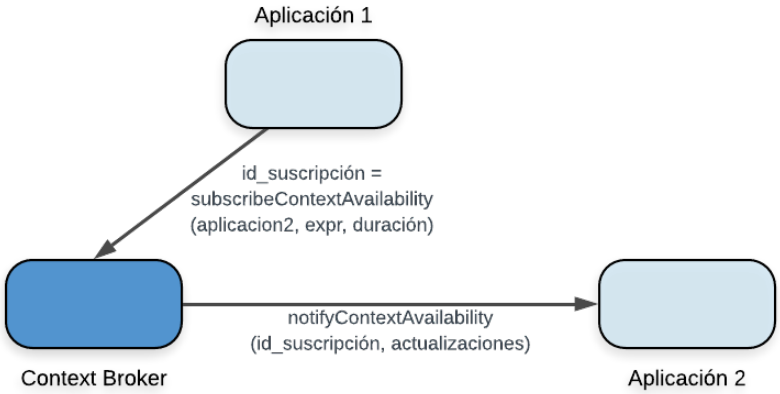

Ilustración 21: Interacciones relacionadas con la suscripción al registro de entidades y atributos

Las aplicaciones que se suscriben a estos eventos recibirán de manera espontánea actualizaciones sobre nuevas entidades o atributos que cumplan determinadas condiciones a través de la operación *notifyContextAvailability*.

5 ARQUITECTURA Y COMPONENTES

En este capítulo se expone el escenario implementado para la realización de este proyecto y los dispositivos y entidades que lo componen. Se describen los dispositivos empleados para la recogida de datos y los habilitadores de la plataforma FIWARE utilizados para el procesado y almacenamiento de los mismos.

Tal y como se adelantaba en el capítulo 1, el objetivo del presente proyecto era realizar una aplicación simple con el fin de explotar las posibilidades que ofrece la tecnología inalábrica LoRa junto con la plataforma FIWARE.

La aplicación finalmente escogida resultó ser una especie de módulo GPS, cuyos datos se gestionan gracias al Context Broker y el resto de habilitadores FIWARE, y pueden ser consultados desde una aplicación. Dicha aplicación es una sencilla página web.

5.1 Escenario planteado

El escenario planteado se compone de tres partes claramente diferenciadas: por un lado, la red de sensores formada por dispositivos físicos, encargada de tomar los datos de localización. Por otro lado, una serie de entidades lógicas FIWARE que realizarán el manejo de los datos de contexto y guardarán el histórico de los cambios producidos en los mismos. La tercera parte es una sencilla aplicación web que consultará la plataforma FIWARE y mostrará los datos recogidos.

La figura 22 muestra un pequeño esquema del escenario. La red de sensores se comunica con la plataforma FIWARE haciendo una petición HTTP POST con los datos recogidos al Context Broker. Para obtener los datos de contexto, la aplicación web realizará la operación contraria: una petición GET para acceder a los mismos.

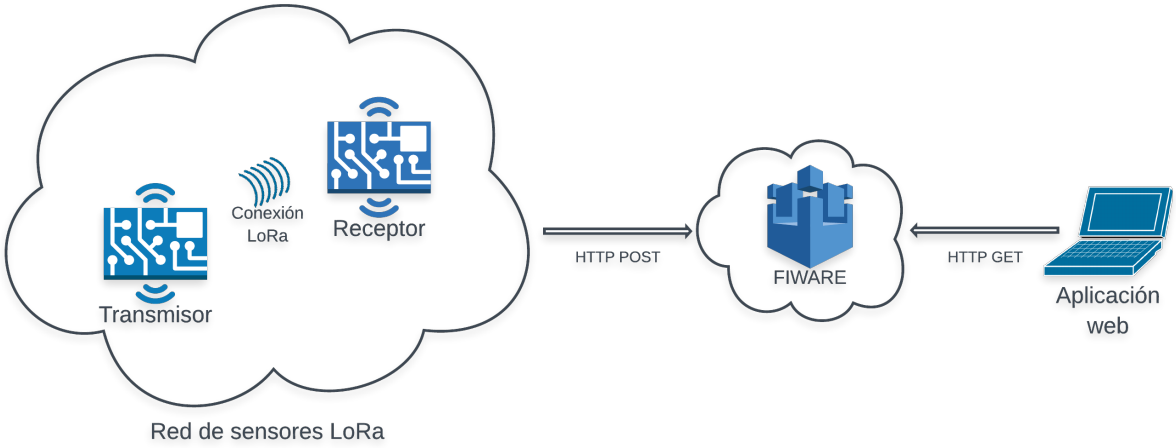

Ilustración 22: Escenario planteado

5.2 La red de sensores Lora

La red de sensores diseñada es bastante sencilla. Consta únicamente de dos nodos:

- El primero de ellos es el sensor IoT. Se trata del nodo transmisor, que recogerá los datos de localización y los enviará al otro nodo, el receptor. La utilidad de este nodo carecería de sentido si estuviera emplazado en un lugar fijo, por lo que para la realización del proyecto se ha colocado sobre un automóvil y se ha alimentado mediante una batería externa. La recogida de datos la realiza a través de un módulo GPS conectado a él. Está alimentado a través de una batería, de manera que no tiene que estar conectado a un ordenador, lo que permite que esté en movimiento.
- El segundo nodo hace el papel de agente IoT. Se trata del nodo receptor, que recibirá los datos recogidos por el transmisor y los almacenará en la plataforma FIWARE, comunicándose para ello con el Context Broker. Para poder realizar esta tarea deberá tener conexión a internet a fin de realizar las peticiones HTTP necesarias. Esta comunicación la hace a través de una conexión WiFi a la IP donde se aloja el servidor del Context Broker. Al contrario que el nodo transmisor, este nodo permanece fijo en un emplazamiento; en este caso conectado a un ordenador, ya que siempre permanecerá escuchando y esperando la recepción de mensajes.

Ambos nodos se comunicarán entre sí mediante la tecnología LoRa. Las placas escogidas para la realización de este montaje, pueden comunicarse a través de LoRa, y también implementan la tecnología WiFi (ideal para la parte de comunicación con el Context Broker).

5.2.1 Placas Heltec Wifi Lora 32

Las placas escogidas para la realización de este proyecto son las Wifi Lora 32 de Heltec Automation [9], una compañía tecnológica especializada en dar soporte a soluciones LoRa.

Las placas de Heltec son compatibles con el entorno de desarrollo de Arduino y están compuestas por varios elementos ya integrados en ella, que facilitan el montaje y desarrollo de los dispositivos.

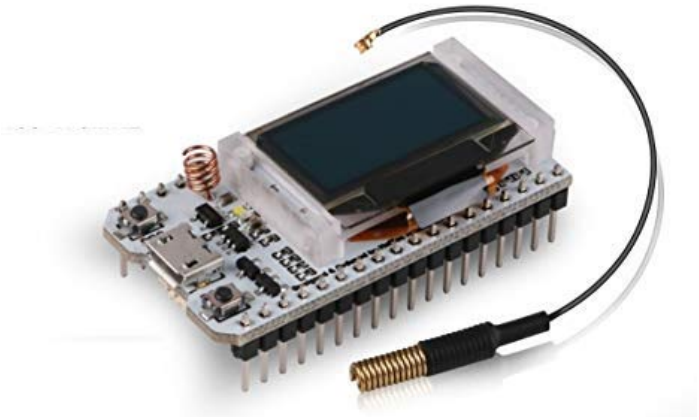

Ilustración 23: Placa Wifi Lora 32 de Heltec

Como veremos a continuación, son placas ideales para el desarrollo de aplicaciones de este tipo y ofrecen muchas posibilidades, ya que en el mismo dispositivo integra varias tecnologías de comunicación y otras características bastante atractivas a la hora de experimentar por primera vez en aplicaciones como la que se va a desarrollar. Además, son bastante económicas.

5.2.1.1 Especificaciones generales de la placa Wifi Lora 32 de Heltec

En la siguiente tabla se recogen los datos más relevantes sobre las especificaciones de la placa Heltec, extraídas de su página web oficial.

Microcontrolador		ESP32
Frecuencia de operación		240MHz
Memora Flash		64MBits
Distancia de comunicación		Hasta 3KM
Potencia de cáculo		Hasta 600dmips
Entorno de desarrollo		Compatible con la suite de Arduino
Voltaje de trabajo		Entre 3,3V y 7V
Rango de temperaturas		-40 – 80℃
Sensibilidad del receptor		-139dbm
Chip convertidor USB – Serial		CP2102
Pantalla OLED	Tamaño	0,96 pulgadas
	Driver	SD1306
	Resolución	128x64 píxeles
Tecnologías soportadas	Wifi	Modos sniffer, Station, softAP
	Lora	Chip SX1276/77/78/79. A escoger: banda de 923MHz o banda de 510MHz
	Bluetooth	Modo dual: Bluetooth tradicional y BLE de bajo consumo
Dimensiones		52 x 25,4 x 10,3 mm

Tabla 9: Especificaciones generales de la placa Wifi Lora 32

Las imágenes 24 y 25 muestran las caras superior e inferior de la placa de Heltec, indicando la posición de sus principales componentes:

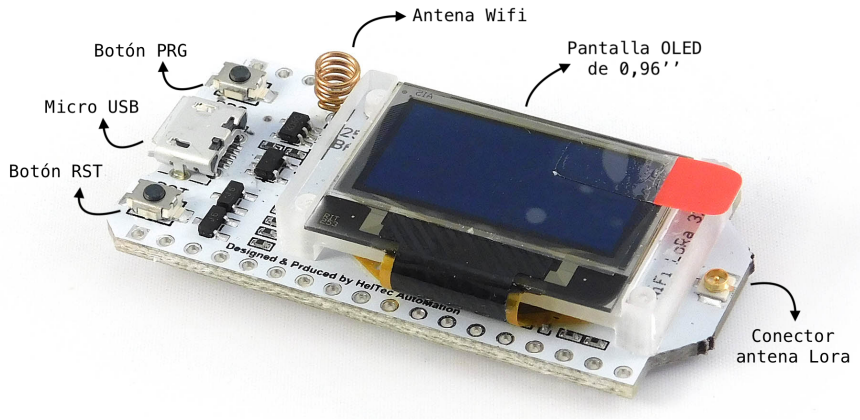

Ilustración 24: Parte superior de la placa Wifi Lora 32

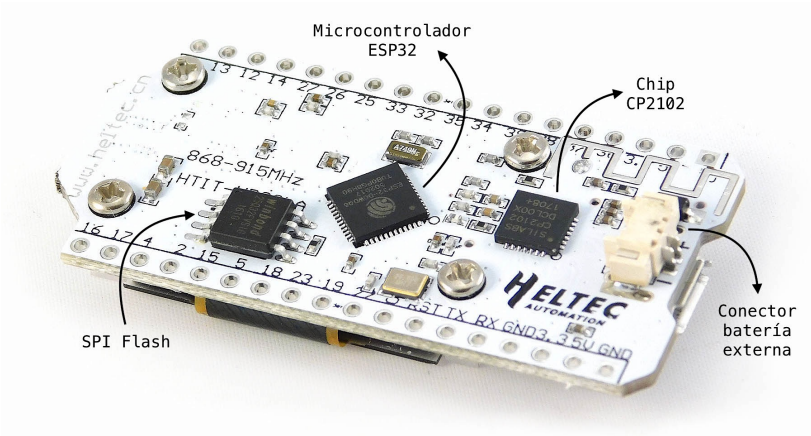

Ilustración 25: Parte posterior de la placa Wifi Lora 32

5.2.1.2 El módulo transceptor Semtech SX1276/77/78/79

La tabla 10 recoge los principales datos sobre el consumo energético del módulo. Es necesario que el dispositivo transmisor implemente métodos para ahorrar energía, por lo que los valores de esta tabla resultan de gran interés.

Modo Sleep	0,2µA
Modo Idle	1,5µA
Modo Standby	1,6 – 1,8µA
Modo sintetizador	5,8mA
Modo recepción	10,8 – 12,0mA
Modo transmisión	20 – 120mA

Tabla 10: Datos de consumo energético del transceptor Lora

5.2.2 Módulo GPS NEO 6M

Para la recogida de datos de localización se empleará el módulo NEO 6M [10] de la compañía suiza UBlox [11]. Se trata de un módulo de tamaño reducido, lo que lo hace ideal para aplicaciones relacionadas con el internet de las cosas, ya que los nodos tienen restricciones de tamaño.

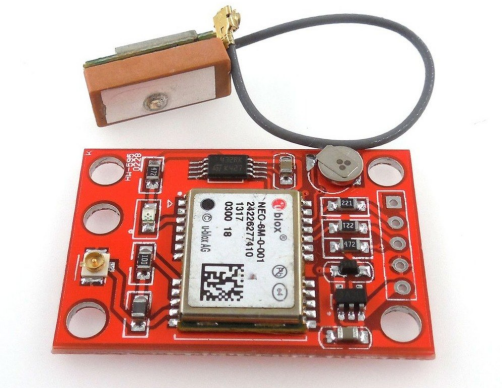

Ilustración 26: Módulo GPS NEO 6M

Este módulo, conectado a la placa Heltec a través de su puerto serial, proporcionará los datos de altitud, latitud longitud y velocidad que serán posteriormente tratados por el Orion Context Broker.

5.2.2.1 Especificaciones del módulo GPS NEO 6M

En la tabla 11 se muestran las principales características y especificaciones del módulo GPS NEO 6M utilizado:

Sensibilidad	Tracking y navegación	-161dBm
	Readquisición	-160dBm
	Inicialización en frío	-147dBm
	Inicialización en caliente	-156dBm
Máxima tasa de actualiación de navegación		5Hz
Precisión posición horizontal		2,5m
Precisión velocidad		0,1m/s
Precisión grados		0,5 grados
Límites operacionales	Altitud	50.000m
	Velocidad	500m/s
Ganancia de la antena		15dB – 50dB
Voltaje de alimentación		-0,5V – 2V

Tabla 11: Especificaciones generales del módulo GPS NEO 6M

5.2.3 Montaje de elementos

Las siguientes imágenes muestran el montaje final de ambos nodos. En la figura 27 se muestra el nodo receptor, conectado a un ordenador ya que se trata del nodo fijo. En las figuras 28 y 29 se muestra el nodo transmisor conectado al módulo GPS.

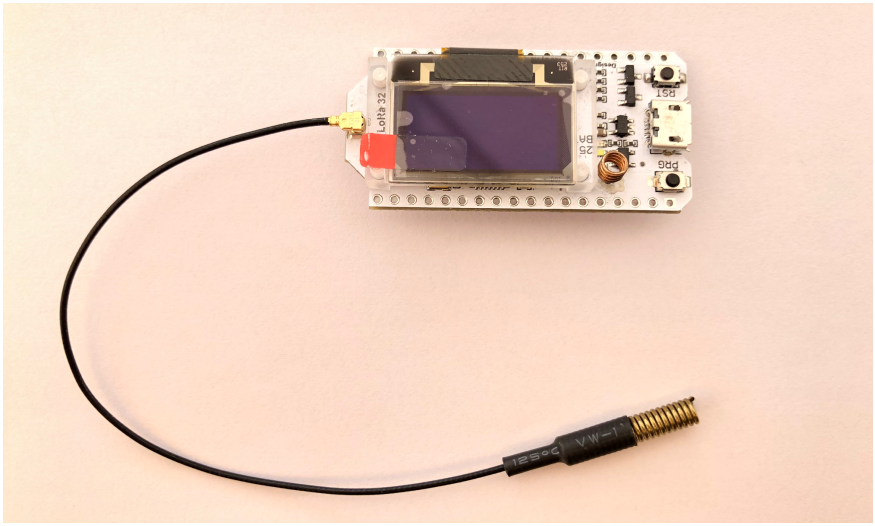

Ilustración 27: Nodo receptor

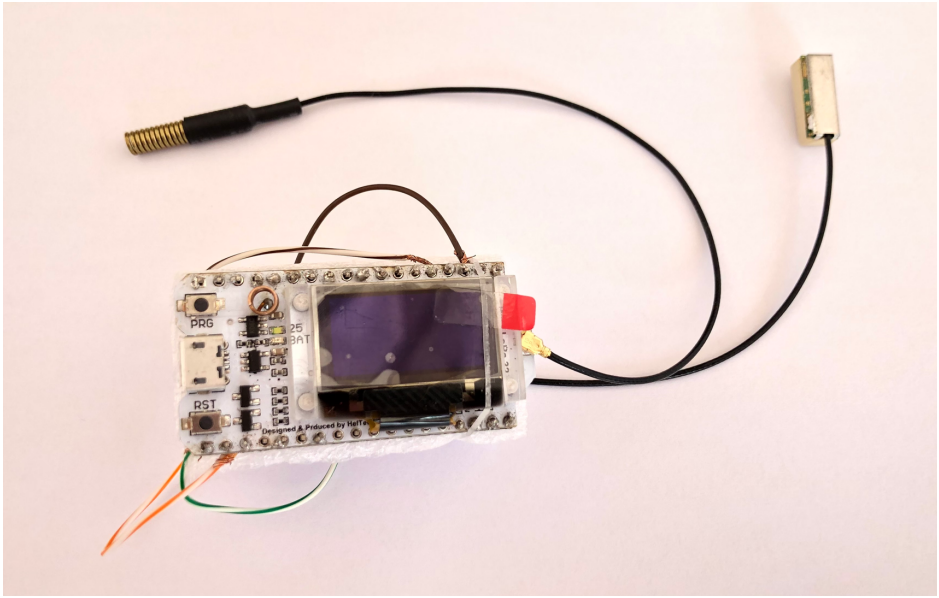

Ilustración 28: Nodo transmisor. Parte superior

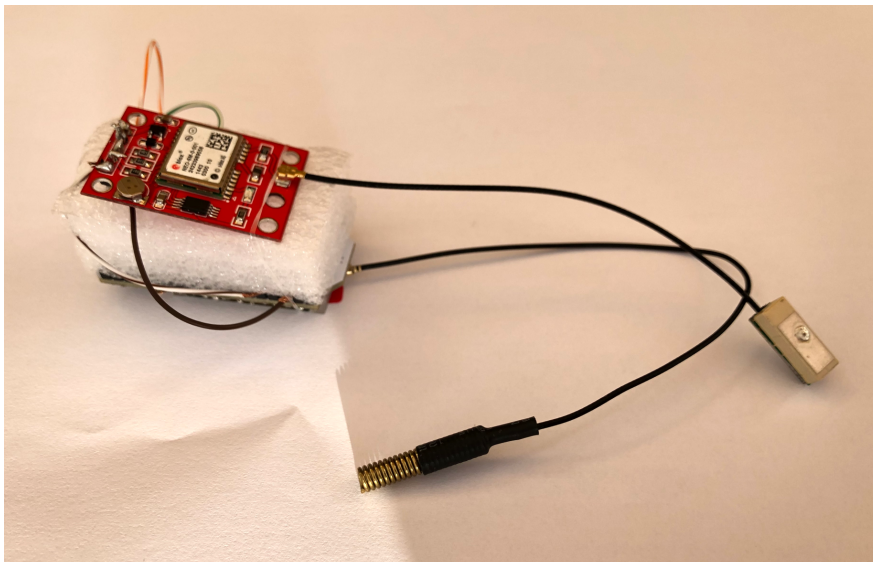

Ilustración 29: Nodo transmisor. Parte posterior

5.3 La aplicación FIWARE

A continuación, se detallan los componentes FIWARE utilizados en la aplicación desarrollada junto al componente principal, el Contex Broker.

5.3.1 El enabler Cygnus

Como ya se ha explicado, el Context Broker solo puede disponer de la información del contexto actual. Para convervar el histórico de cambios en el contexto se utiliza el conector Cygnus [12], aportando así persistencia de los mismos.

Internamente, Cygnus se basa en Apache Flume [13], un servicio distribuido y abierto que permite recopilar, agregar y mover de manera eficiente un gran volumen de datos de contexto. Su arquitectura se basa en flujos de transmisión y es bastante simple y flexible.

Los conceptos que maneja Flume son:

- **Evento**. Volumen de datos que Flume puede transportar desde su punto de origen hasta su punto final.
- **Flujo**. Movimiento de eventos desde el punto de inicio al punto final.
- **Cliente**. Entidad que opera en el punto de origen de los eventos y los entrega al agente Flume.
- **Agente**. Proceso independiente que aloja al resto de componentes Flume, que tiene la capacidad de recibir, almacenar y reenviar eventos hasta su próximo destino. Los componentes de Fluje son:
    - **Fuente o source**. Entidad que puede consumir eventos entregados a ella a través de cierto mecanismo. Cuando una fuente recibe un evento, lo entrega a uno o más canales.
    - **Canal o channel**. Almacenamiento temporal de los eventos. Los eventos depositados en un canal permanecen en él hasta que un sumidero los elimina. Representa un papel crucial respecto a la durabilidad de los flujos.
    - **Sumidero o sink**. Entidad que puede eliminar eventos de un canal y transmitirlos al siguiente agente en el flujo o hasta el destino final del evento.

La figura 30 representa la arquitectura explicada en la que se basa Apache Flume.

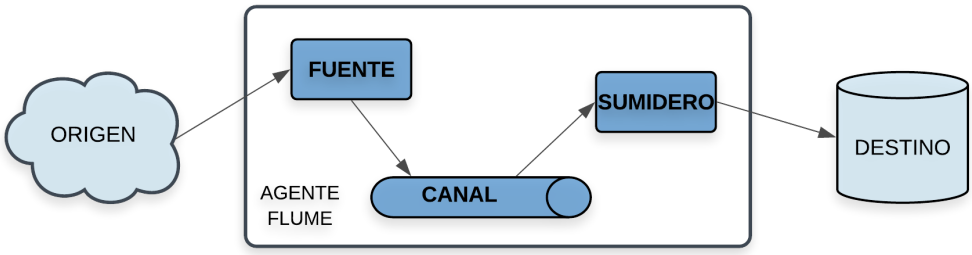

Ilustración 30: Arquitectura de Apache Flume

Cygnus está diseñado para hacer funcionar una instancia de Apache Flume por cada fuente de datos a la que se conecte. Como destino de los datos, pueden configurarse numerosas plataformas de terceros compatibles con Cygnus. Además de la que se utiliza en este proyecto, mongo DB, también pueden emplearse otras bases de datos como Postgree, sistemas distribuidos de ficheros como HDFS de Hadoop o otros conectores FIWARE, entre otros.

En la arquitectura FIWARE, Cygnus juega el papel de conector entre el Orion Context Broker, el cual haría el papel de fuente de datos, y otros muchos elementos de almacenamiento FIWARE como son Sth-comet (utilizado en este proyecto), CKAN o Cosmos Big Data.

5.3.2 El enabler STH-Comet

El "Short Time Historic" o STH-Comet [14] es un componente del ecosistema FIWARE que permite administrar (almacenar y recuperar) información de contexto histórica sin procesar, además de procesar información agregada sobre la evolución de los datos de contexto registrados por el Context Broker a lo largo del tiempo.

Este componente permite almacenar y recolectar la información sobre los cambios de contexto sin formato, es decir, permite obtener el valor del atributo que se registró en cierto momento en el Context Broker. Sin embargo, su mayor potencial es la capacidad de generar series de información de contexto agregada a lo largo del tiempo. Para ello, gestiona cuatro conceptos básicos:

- **Resolución o periodo de agregación**. Se trata del periodo de tiempo en el que los datos de contexto serán agrupados para conformar las series agregadas. Los posibles valores de resolución son: mes, día, hora, minuto o segundo.
- **Origen**. Para una cierta resolución, se trata del origen de tiempo al que aplicar las series de datos.
- **Offset**. Para una determinada resolución, indica el desplazamiento con respecto el origen para el que se aplica la agregación de la información.
- **Samples**. Para una resolución, origen y offset en concreto, se trata del número de muestras, valores, eventos o notificaciones disponibles desde el origen.

STH-Comet dispone de dos modos de funcionamiento: el modo minimalista y el modo formal.

- **Modo minimalista**. Solo se emplea el componente Sth-Comet, encargado de almacenar los datos históricos en MongoDB y también de leerlos. Este modo es mucho más sencillo de configurar, ya que el componente Cygnus no intervendría en el escenario. Por el contrario, no puede configurarse para trabajar con otras bases de datos que no sean MongoDB ni tampoco resulta muy escalable ni válido para aplicaciones con un flujo de datos importante.
- **Modo formal**. El componente Sth-Comet se utiliza junto a Cygnus, siendo éste el encargado de

recolectar los datos históricos, y Sth-Comet de leerlos. Este modo es altamente configurable y ofrece más posibilidades en escenarios complejos. Para hacer posible la recolección de datos en combinación con Cygnus se implementan dos sumideros:

- **El sumidero de mongo DB**. Se encarga de la persistencia de los datos de contexto tal y como son registrados en el Orion Context Broker. Una vez almacenados en esta base de datos, STH-Comet puede hacer disponible esta información a través de su API de información de contexto sin procesar.
- **El sumidero de STH**. Se encarga de persistir en la base de datos la información de contexto relativa a las series de tiempo deseadas. La información es agregada previamente con las resoluciones que se hayan configurado y no en el momento de la recuperación de los datos, de manera que consultar la información agregada almacenada sea algo instantáneo.

Aunque la aplicación a desarrollar en este proyecto no utiliza un escenario complejo, se ha optado por implementar el modo formal de cara a desarrollar futuras mejoras.

Es importante indicar que la duración de las suscripciones realizadas a STH-Comet tienen una cierta duración, tras la cual la suscripción expira y es necesario volver a realizarla.

6 SOFTWARE E IMPLEMENTACIÓN

Para la realización de esta aplicación se han utilizado diversas tecnologías, funcionando todas juntas para lograr un propósito. En este capítulo se detalla la parte software del proyecto, es decir, la programación de los nodos para la red de sensores LoRa, la implementación de los componentes Fiware y el desarrollo de la aplicación web.

La aplicación Fiware se ha realizado de dos formas con el fin de experimentar con el entorno. Por una parte, se ha desarrollado la aplicación FIWARE usando la tecnología Docker. Por otro lado, se ha probado el servicio que FIWARE pone a disposición de sus usuarios: FIWARE Lab. Se trata de una plataforma en la que ofrecen un entorno de trabajo y una serie de recursos para que el usuario pueda desarrollar sus proyectos.

En las siguientes secciones se expone la configuración y despliegue de los elementos anteriormente citados.

6.1 Implementación de la red de sensores LoRa

En las siguientes secciones se expone todo lo relacionado con el desarrollo de la red de sensores que se comunican a través de la tecnología LoRa.

6.1.1 Entorno de trabajo

Para programar los dispositivos Heltec Lora Wifi 32 que se emplean en este proyecto, puede utilizarse el entorno de programación o IDE de Arduino [15], aunque para ello deben instalarse una serie de drivers y librerías que posibilitan su funcionamiento. Cabe destacar que, aunque se programan de la misma forma, se han encontrado numerosas trabas para lograr un correcto funcionamiento, ya que los dispositivos no respondían de la misma forma al usar algunas funciones.

El IDE o Integrated Development Environment es un entorno de programación compuesto por diferentes herramientas que ayudan a desarrollar aplicaciones. Se compone usualmente de un editor de código, un compilador, un depurador y una interfaz gráfica.

El entorno de desarrollo de Arduino incorpora barra de herramientas, editor de código, terminal de mensajes del compilador y monitor serie, entre otras funcionalidades.

La programación de los dispositivos Heltec es similar a la de las placas Arduino, por lo que hereda el lenguaje y todas las características asociadas. El software escrito para Arduino se denomina sketch, está basado en C y C++ y librerías estándar de este lenguaje. Los sketches son convertidos a lenguje máquina a través de un compilador tipo gcc propio de los microprocesadores Atmel que incorporan las placas. Este

código ya convertido, combinado con las librerías básicas de Arduino, contiene el fichero binario que será cargado en la memoria de programa del chip de la placa.

La estructura principal de un sketch en Arduino y, por tanto, en las placas Heltec, se compone de dos funciones principales: `setup()` y `loop()`. `Setup()` es la función llamada cuando comienza la ejecución del sketch y contiene todas las inicializaciones de variables, inicio de módulos de la placa mediante librería y seteo de pines como entrada/salida o encendido/apagado. Esta función solo se ejecuta una vez, después de cada reseteo o encendido de la placa.

`Loop()` es, como su propio nombre indica, el bucle que se mantiene en ejecución y provee un control active de la placa, provocando que ésta responda ante cambios externos según se programe.
Las estructuras de control, operadores lógicos, sintaxis, formatos de variables y funciones básicas son idénticas a las de C/C++.

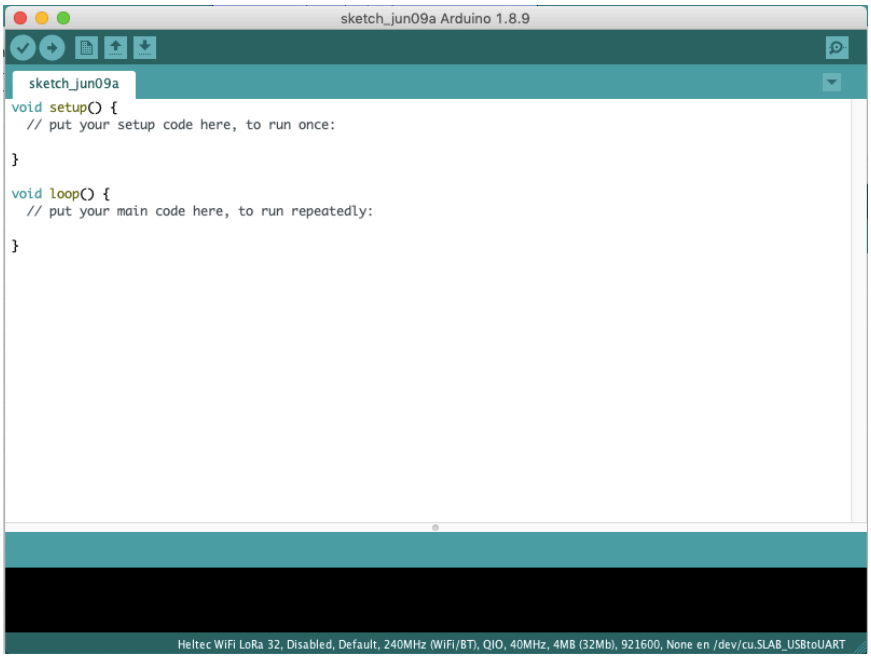

Ilustración 31: Entorno de programación de Arduino

Los drivers y librerías necesarios para trabajar con las placas son los siguientes:

- Drivers para el chip ESP32 que poseen las placas Heltec Lora Wifi 32. Pueden encontrarse en la web del proveedor, o bien instalarlos ejecutando el siguiente comando (en el caso del sistema operativo macOS):

```
mkdir -p ~/Documents/Arduino/hardware/espressif && \
cd ~/Documents/Arduino/hardware/espressif && \
git clone https://github.com/espressif/arduino-esp32.git esp32
&& \
cd esp32 && \
git submodule update --init --recursive && \
cd tools && \
python get.py
```

- Librería `u8g2`, para las pantallas de los dispositivos. Esta funcionalidad se ha utilizado durante las

pruebas, ya que resultaba más cómodo ver resultados en pantalla. Para el funcionamiento final se ha eliminado su uso debido a que mantener la pantalla encendida consume mucha batería para mantener la red de sensores.

- Librería `TinyGPS++`, para poder hacer uso del GPS instalado en el nodo transmisor.
- Librería `LoRa`, para programar la comunicación utilizando la tecnología Lora entre ambos dispositivos.
- Librería `WiFi`, para poder conectar el dispositivo a una red de internet a través de la cual poder comunicarse con el Context Broker.
- Librería `HTTPClient`, para realizar peticiones HTTP y poder registrar los datos medidos en el Context Broker.

6.1.2 Diagrama de flujo de los dispositivos

La siguiente imagen muestra el diagrama de flujo del nodo transmisor, el cual recoge datos de localización y los envía mediante LoRa al nodo receptor.

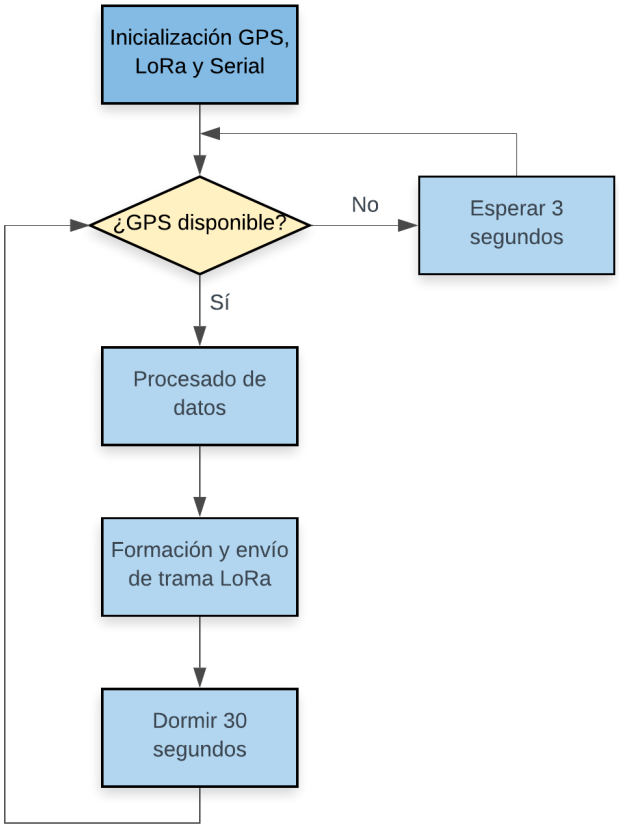

Ilustración 32: Diagrama de flujo del nodo transmisor

La imagen 33 representa el diagrama de flujo del nodo receptor, el cual permanece escuchando y cuando recibe un paquete LoRa extrae los datos necesarios para registrarlos en el Context Broker a través de una petición HTTP POST.

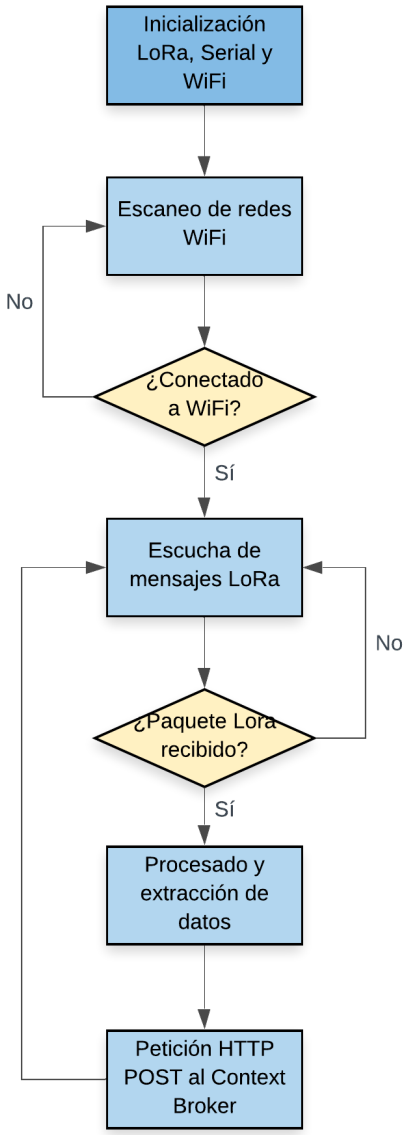

Ilustración 33: Diagrama de flujo del nodo receptor

6.2 Desarrollo y configuración de la aplicación FIWARE

6.2.1 Escenario

En la figura 34 se muestra la arquitectura completa de la aplicación FIWARE:

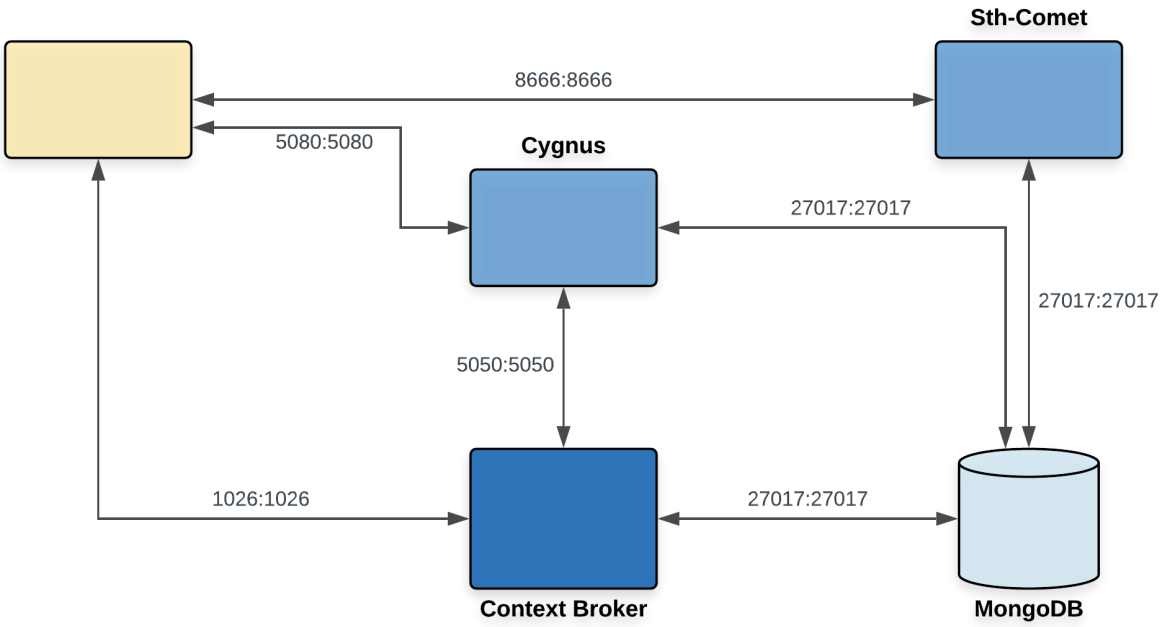

Ilustración 34: Esquema de componentes de la aplicación FIWARE

El escenario cuenta con los siguientes componentes:

- El componente principal y esencial en cualquier aplicación impulsada por FIWARE es, como ya se ha comentado, el Orion Context Broker. Este componente escucha peticiones HTTP en el puerto 1026 usando NGSI.
- La base de datos de código abierto Mongo DB (aunque pueden configurarse otras bases de datos) se emplea para almacenar toda la información necesaria relativa a entidades, datos de contexto, suscripciones y registros. Se ha configurado MongoDB para que escuche en el puerto 27017.
- Como ya se comentó en el capítulo 4 de este documento, el Context Broker combinado con MongoDB solo puede almacenar la información de contexto más actual de una entiadad, por lo que, para disponer de un histórico de la misma, es necesario introducir en el escenario el componente Cygnus. Este habilitador se configurará para recibir suscripciones a los cambios en la información de contexto que maneja nuestra aplicación y permitirá disponer de un historial de localizaciones recogidas por la red de sensores, logrando así la persistencia de los datos.
- El componente Sth-Comet permite leer los datos históricos almanceados en la base de datos. Además, permite realizar consultas sobre datos agregados, como la media de un cierto valor, valores máximos y mínimos, consultas por tiempos, etc. Tal y como se introdujo en el capítulo 4, este componente se configurará en modo formal a pesar de no ser de necesidad, con vistas a posibles mejoras futuras.

6.2.2 Docker e instalación de componentes

Para facilitar el despliegue de la aplicación FIWARE, se ha utilizado la plataforma de virtualización Docker [16], la cual ha permitido prescindir de la preparación de un entorno de trabajo instalando software adicional (como, por ejemplo, una máquina virtual). De esta forma, todos los elementos de la aplicación ya se encuentran contenidos dentro de imágenes virtuales proveídas por el repositorio de Docker.

A continuación, se muestran una serie de comandos empleados para la descarga y configuración de los elementos FIWARE. Se muestra solo un ejemplo de cómo configurar el Context Broker con la base de datos Mongo DB, ya que la implementación completa del escenario se ha realizado mediante una herramienta más cómoda.

Para descargar las imágenes de los enablers Fiware se emplean el comando `docker pull`, indicando la imagen a descargar y su versión.

```
docker pull mongo:3.6
docker pull fiware/orion
```

Se debe crear una red que aloje todos los componentes anteriores. Esta operación se realiza mediante el comando `docker network create`.

```
docker network create fiware_default
```

Finalmente, cada componente es conectado a la red anteriormente creada, utilizando los siguientes comandos:

```
docker run -d --name=mongo-db --network=fiware_default \
  --expose=27017 mongo:3.6 --bind_ip_all -smallfiles

docker run -d --name fiware-orion -h orion --network=fiware_default -p
1026:1026  fiware/orion -dbhost mongo-db
```

Como puede apreciarse, no se indica el número de los puertos en los que escucha cada componente, ya que los valores utilizados en esta aplicación son los que usan por defecto.

Además de los comandos anteriores, pueden utilizarse los siguientes para realizar funciones como parar los contenedores, borrarlos, o borrar la red creada.

```
docker stop fiware-orion
docker rm fiware-orion
docker stop mongo-db
docker rm mongo-db
docker network rm fiware_default
```

Para facilitar todo el proceso de instalación y configuración, se ha utilizado `docker-compose`, una herramienta muy útil para desplegar aplicaciones que hacen uso de varios contenedores diferentes. En este caso, cada enabler FIWARE es un contenedor distinto, por lo que ha resultado de gran utilidad el uso de esta herramienta. En lugar de descargar todas las imágenes de los contenedores manualmente, se crea un fichero YAML en el que se configuran todos los contenedores. Este fichero se ejecuta en distintos modos según se quieran descargar las imágenes, inicializar los contenedores o parar el servicio y borrar los componentes. Para su llamada se hace uso de otro fichero denominado “services”.

A continuación, se detallan las partes más relevantes del fichero de configuración.

En el siguiente extracto de código del fichero `docker-compose.yml` se configura el Context Broker. Se especifica que su funcionamiento depende del elemento Mongo-db, la red en la que va a implantarse y el puerto en el que escucha.

```
orion:
    image: fiware/orion:2.2.0
```

```
    hostname: orion
    container_name: fiware-orion
    depends_on:
      - mongo-db
    networks:
      - default
    expose:
      - "1026"
    ports:
      - "1026:1026"
    command: -dbhost mongo-db -logLevel DEBUG
    healthcheck:
      test: curl --fail -s http://orion:1026/version || exit 1
```

En el siguiente extracto de código se configura el elemento Cygnus. Se especifica que su funcionamiento depende del elemento Mongo-db, la red en la que va a implantarse y los puertos en los que escucha. Uno de ellos se ha configurado solo para razones de gestión.

```
cygnus:
    image: fiware/cygnus-ngsi:1.10.0
    hostname: cygnus
    container_name: fiware-cygnus
    depends_on:
      - mongo-db
    networks:
      - default
    expose:
      - "5050"
      - "5080"
    ports:
      - "5050:5050"
      - "5080:5080"
    environment:
      # Servidor en el que almacenar el histórico de contexto
      - "CYGNUS_MONGO_HOSTS=mongo-db:27017"
      # Nivel de log
      - "CYGNUS_LOG_LEVEL=DEBUG"
      # Puerto de escucha para la suscripción a cambios de contexto
      - "CYGNUS_SERVICE_PORT=5050"
      # Puerto de escucha para razones operacionales únicamente
      - "CYGNUS_API_PORT=5080"
    healthcheck:
      test: curl --fail -s http://localhost:5080/v1/version || exit
```

En el siguiente extracto de código se configura el elemento Sth-comet. Se especifica que depende tanto de Cygnus como de Mongo DB, el puerto en el que escucha, y el prefijo del nombre de la base de datos que se creará para el almacenamiento del histórico.

```
sth-comet:
    image: fiware/sth-comet:2.5.0
    hostname: sth-comet
    container_name: fiware-sth-comet
    depends_on:
      - cygnus
      - mongo-db
```

```
    networks:
      - default
    ports:
      - "8666:8666"
    environment:
      # IP y puerto
      - STH_HOST=0.0.0.0
      - STH_PORT=8666
      # Prefijo del nombre de la base de datos creada
      - DB_PREFIX=sth_
      # URI para la comunicación con mongoDB
      - DB_URI=mongo-db:27017
      # Nivel de log
      - LOGOPS_LEVEL=DEBUG
    healthcheck:
      test: curl --fail -s http://localhost:8666/version || exit 1
```

Por último, se configura la base de datos Mongo.

```
  mongo-db:
    image: mongo:3.6
    hostname: mongo-db
    container_name: db-mongo
    expose:
      - "27017"
    ports:
      - "27017:27017"
    networks:
      - default
    command: --bind_ip_all --smallfiles
    volumes:
      - mongo-db:/data
```

A continuación, se muestra el contenido del script `services`, a través del cual se descargan las imágenes, se inicializan, o se para el servicio. Se puede ejecutar con las opciones `create`, `start` o `stop`, según la acción se quiera realizar.

```
#!/bin/bash

#  Interfaz por línea de comandos para iniciar todos los servicios asociados
a la aplicación
#
#  Gloria Martínez Muñoz
#

set -e

if (( $# != 1 )); then
        echo "Error en el número de parámetros"
        echo "Uso: services [create|start|stop]"
        exit 1
fi

stoppingContainers () {
        echo "Parando contenedores"
        docker-compose --log-level ERROR -p fiware down -v --remove-orphans
}
```

```
displayServices () {
        echo ""
        docker ps --format "table {{.Names}}\t{{.Status}}\t{{.Ports}}" --
filter name=fiware-*
        echo ""
}

command="$1"
case "${command}" in
        "help")
                echo "Uso: services [create|start|stop]"
                ;;
        "start")
                stoppingContainers
                echo -e "Inicializando los cuatro contenedores:
\033[1;34mOrion\033[0m, \033[1;34mCygnus\033[0m,\033[1;34mSth-comet\033[0m y
la base de datos \033[1;34mMongoDB\033[0m."
                echo -e "- \033[1;34mOrion\033[0m es el context broker."
        echo -e "- \033[1;34mCygnus\033[0m únicamente configurado para
escribir información de contexto en Mongo DB."
        echo -e "- \033[1;34mSth-Comet\033[0m configurado para poder
realizar consultas basadas en tiempos"
        echo -e "- \033[1;34mMongo-DB\033[0m es la base de datos donde se
apoya cada elemento anterior"
                echo ""
                docker-compose --log-level ERROR -p fiware up -d --remove-
orphans
        displayServices
                ;;
        "stop")
                stoppingContainers
                ;;
        "create")
        echo "Obtención de las imágenes docker"
                docker-compose --log-level ERROR -f docker-compose.yml -p
fiware pull
                ;;
        *)
                echo "Comando no encontrado."
                echo "Uso: services [create|start|stop]"
                exit 127;
                ;;
esac
```

La figura 35 muestra la ejecución de los ficheros anteriormente citados para crear las imágenes y poner en funcionamiento el servicio FIWARE diseñado. Cuando el proceso termina, se muestra un resumen con los contenedores creados, su estado y los puertos que tienen abiertos.

```
[MacBook-Air-de-Gloria:TFM gloria$ ./services create                                ]
Obtención de las imágenes docker
Pulling mongo-db  ... done
Pulling cygnus    ... done
Pulling sth-comet ... done
Pulling orion     ... done
[MacBook-Air-de-Gloria:TFM gloria$ ./services start                                 ]
Parando contenedores
Inicializando los cuatro contenedores: Orion, Cygnus,Sth-comet y la base de datos MongoDB.
- Orion es el context broker.
- Cygnus únicamente configurado para escribir información de contexto en Mongo DB.
- Sth-Comet configurado para poder realizar consultas basadas en tiempos
- Mongo-DB es la base de datos donde se apoya cada elemento anterior

Creating db-mongo ... done
Creating fiware-cygnus ... done
Creating fiware-orion  ... done
Creating fiware-sth-comet ... done

NAMES               STATUS                                  PORTS
fiware-sth-comet    Up Less than a second (health: starting)   0.0.0.0:8666->8666/tcp
fiware-cygnus       Up 1 second (health: starting)             0.0.0.0:5050->5050/tcp, 0.0.0.
0:5080->5080/tcp
fiware-orion        Up 1 second (health: starting)             0.0.0.0:1026->1026/tcp

MacBook-Air-de-Gloria:TFM gloria$
```

Ilustración 35: Creación y puesta en marcha del servicio FIWARE

El resultado de parar la aplicación fiware, deteniendo y borrando los contenedores anteriormente creados, se muestra en la figura 36.

```
[MacBook-Air-de-Gloria:TFM gloria$ ./services stop                                  ]
Parando contenedores
Stopping fiware-sth-comet ... done
Stopping fiware-cygnus    ... done
Stopping fiware-orion     ... done
Stopping db-mongo         ... done
Removing fiware-sth-comet ... done
Removing fiware-cygnus    ... done
Removing fiware-orion     ... done
Removing db-mongo         ... done
MacBook-Air-de-Gloria:TFM gloria$
```

Ilustración 36: Parada de la aplicación Fiware y borrado de contenedores

6.2.3 Operaciones realizadas

Todas las peticiones que se detallan en este apartado se han realizado a través de la herramienta cURL [17]. Se trata de una herramienta orientada a la transferencia de archivos, compuesta por una librería (libcurl) y un intérprete de comandos. Con ella se pueden simular peticiones HTTP.

6.2.3.1 Creación de entidades

Para el escenario de esta aplicación, solo ha sido nececaria la creación del elemento Device, el cual representa al dispositivo LoRa que envía a la plataforma FIWARE los datos de localización. La entidad posee varios atributos:

- Categoría del dispositivo: se trata de un sensor.
- Propiedad a controlar: datos de localización, los cuales se recogerán en formato GEO Json [18].
- Protocolo soportado por el dispositivo: la tecnología LoRa.
- El nivel de batería del dispositivo.
- El atributo `value` hace referencia a la altura del dispositivo.

El JSON empleado para su definición, siguiendo la normal establecida, es el siguiente:

```
  {
      "type": "Device",
      "id": "urn:ngsi-ld:Device:001",
      "category": {
          "type": "text",
          "value": "sensor"
      },
       "controlledProperty": {
          "type": "text",
          "value": "location"
      },
       "supportedProtocol": {
          "type": "text",
          "value": "lora"
      },
      "location": {
          "type": "geo:json",
          "value": {
               "type": "Point",
               "coordinates": [0, 0]
          }
      },
       "batteryLevel": {
          "type": "number",
          "value": "0"
      },
       "value": {
          "type": "number",
          "value": "0"
      }
  }
```

6.2.3.2 Suscripción a cambios de contexto

Se han realizado suscripciones a los cambios de contexto en los atributos de localización y altura del dispositivo. Habría sido muy interesante realizar una suscripción a los cambios en la batería del dispositivo a fin de realizar pruebas de duración de la misma, pero durante la realización del proyecto se ha comprobado que las placas utilizadas no disponen de ningún método para obtener el nivel de energía de la batería.

Para realizar la suscripción anteriormente descrita, se ejecuta la siguiente petición HTTP, en la que se indica que se desea realizar una suscripción a los cambios de contexto del atributo `location`, para todas las entidades cuyo id coincida con el patrón `urn:ngsi-ld:Device`. Cuando se detecten estos cambios, se informará al puerto 5050 en el que escucha Cygnus.

```
  curl              --location              --request              POST
"http://localhost:1026/v2/subscriptions/" \
    --header "Content-Type: application/json" \
    --data "{
    "description": "Notificar a Cygnus que compruebe los datos de contexto
del atributo localización cada 5 segundos",
    "subject": {
      "entities": [
```

```
        {
          "idPattern": " urn:ngsi-ld:Device.*"
        }
      ],
      "condition": {
        "attrs": [
          "location"
        ]
      }
    },
    "notification": {
     "http": {
        "url": "http://cygnus:5050/notify"
      },
      "attrs": [
        "location"
      ],
      "attrsFormat": "legacy"
    },
    "throttling": 5
  }"
```

Del mismo modo se haría la suscripción a los cambios de contexto del abributo de altura del dispositivo.

Una vez creada la entidad y realizadas las suscripciones, es posible acceder a la base de datos de mongo para ver "las tripas" de la aplicación y comprobar donde se almacena cada elemento de contexto dentro de ella. Como puede apreciarse en la figura 37, accediendo a mongo se aprecia la existencia de varias bases de datos. Las importantes para el desarrollo de la aplicación son:

- **Orion**. En ella se almacenan las definiciones de todas las entidades (en la tabla entities) y todas las suscripciones (en la tabla csubs) así como los datos de contexto del momento actual. NOTA: para la realización del proyecto no se ha especificado el nombre de un servicio en concreto, si no que se ha mantenido por defecto. En caso de haberlo especificado, se crearía otra base de datos con nombre orion-`nombre_servicio`. De esta forma pueden coexistir datos de contexto de múltiples servicios en la misma instancia de mongo DB.
- **Sth_default**. Almacena los datos históricos obtenidos a través de Cygnus para los atributos a los que se ha realizado suscripción. Estos datos serán leídos por el habilitador Sth_comet. Como puede comprobarse, el nombre de la base de datos consta del prefijo configurado para Sth_comet, seguido del nombre del servicio utilidado (default para el caso de esta aplicación). Al igual que en el caso anterior, se crearían tantas bases de datos como servicios en el caso de especificar alguno diferente.

  Accediendo a esta base de datos, puede comprobarse la existencia de dos colecciones distintas. Por un lado, la información relativa al histórico de contexto al que se ha suscrito la entidad y, por otro, la colección de información de datos agregados que sth-comet es capaz de manejar.

```
MacBook-Air-de-Gloria:~ gloria$ docker run -it --network fiware_default  --entrypoint /bin/bash mongo:3.6
root@69674cb3a167:/# mongo --host mongo-db
MongoDB shell version v3.6.13
connecting to: mongodb://mongo-db:27017/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("9b245db7-2cac-4163-8c8a-3a55a3941b2b") }
MongoDB server version: 3.6.13
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2019-06-21T20:40:58.219+0000 I STORAGE  [initandlisten]
2019-06-21T20:40:58.219+0000 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly re
commended with the WiredTiger storage engine
2019-06-21T20:40:58.219+0000 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prod
notes-filesystem
2019-06-21T20:40:59.135+0000 I CONTROL  [initandlisten]
2019-06-21T20:40:59.135+0000 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the d
atabase.
2019-06-21T20:40:59.135+0000 I CONTROL  [initandlisten] **          Read and write access to data and confi
guration is unrestricted.
2019-06-21T20:40:59.135+0000 I CONTROL  [initandlisten]
> show dbs
admin        0.000GB
config       0.000GB
local        0.000GB
orion        0.000GB
sth_default  0.000GB
> use orion
switched to db orion
> show collections
csubs
entities
> use sth_default
switched to db sth_default
> show collections
sth_/_urn:ngsi-ld:Device:001_Device
sth_/_urn:ngsi-ld:Device:001_Device.aggr
>
```

Ilustración 37: Acceso a mongoDB

6.2.4 FIWARE Lab

Durante el desarrollo de este proyecto, también se han querido probar las posibilidades de la plataforma FIWARE Lab [19], un entorno de pruebas no comercial que FIWARE pone a disposición de su comunidad para la experimentación. En ella, Los usuarios pueden probar la tecnología y sus aplicaciones haciendo uso de los recursos proporcionados: la infraestructura de cómputo como servicio (máquinas virtuales, espacio de almacenamiento, servicios de red), los componentes de software de la Plataforma FIWARE y los datos abiertos publicados por ciudades y otras organizaciones, bien sea para utilizarlos como un servicio público o para mantener de forma privada la instancia. FIWARE Lab se conforma de un conjunto de nodos federados distribuidos alrededor del mundo.

Ilustración 38: Logo FIWARE Lab

En el anexo 2 se presentan los pasos que se han seguido para crear una instancia en FIWARE Lab y los primeros pasos que deben seguirse para crear una aplicación impulsada por Fiware en ella.

6.3 Aplicación web para visualizar los datos

Para poder visualizar los datos de contexto registrados por el Context Broker, se ha diseñado una sencilla aplicación web. En esta sección se detallan las tecnologías utilizadas y se muestra el aspecto de la misma, así como las partes claves del código para la comunicación con los elementos Fiware.

6.3.1 Tecnologías utilizadas

6.3.1.1 HTML y CSS

HTML [20] es un lenguaje de programación de marcado empleado para la elaboración de páginas de Internet. Su nombre hace referencia a las siglas HyperText Markup Language, es decir, "Lenguaje de marcas de hipertexto". Debe su nombre a que su utilidad es indicar en qué posición aparecen los elementos de una página web a través de marcas o etiquetas.

Se trata de un estándar a cargo del Consorcio WWW o World Wide Web (W3C) [21], una organización dedicada a la estandarización de prácticamente todas las tecnologías asociadas a la web.

Mediante HTML se definen referencias a la ubicación de los distintos elementos que componen la aplicación web, recayendo en el navegador web o intérprete el papel de unir todos los elementos y visualizar el resultado final.

Por otra parte, CSS (Cascading Style Sheets) es el lenguaje empleado para describir cómo se presentan las páginas web, su apariencia, incluyendo colores, diseño y tipos de fuente. Permite adaptar la presentación a todos los tipos de dispositivo, de manera que la apariencia de la aplicación web se mantenga igual independientemente de dónde se visualice.

CSS no es dependiente de HTML, sino que puede utilizarse con cualquier otro lenguaje de marcado, como por ejemplo XML.

6.3.1.2 PHP

PHP (*Hypertext Preprocessor*) es un lenguaje de programación de propósito general de código abierto, originalmente diseñado para el desarrollo de aplicaciones web con contenido dinámico. Puede incorporarse directamente a un documento HTML o bien llamar a un fichero externo que procese los datos.

Se trata de un lenguaje del lado del servidor, es decir, se ejecuta en el servidor web, justo antes de que se envíe la página web a través de Internet al cliente. Es por ello que el código desarrollado en PHP debe ser interpretado por un servidor web que disponga de un módulo de procesado, para posteriormente generar el HTML resultante.

Las páginas que se ejecutan en el servidor a través de PHP pueden realizar consultas a bases de datos, conexiones en red y otro tipo de tareas para generar la web que visualizará el cliente.

6.3.1.3 Javascript y AJAX

Javascript es un lenguaje de programación que permite crear contenido nuevo y dinámico mediante la realización de actividades complejas dentro de una página web, como mostrar actualizaciones de contenidos en tiempo real, introducir animaciones gráficas, interacturar con mapas, etc. Conforma la tercera capa de la agrupación entre HTML, CSS y Javascript.

Se trata de un lenguaje de programación interpretado y orientado a objetos, utilizado normalmente en el lado del cliente. Su sintaxis es muy parecida a C, con algunas modificaciones en cuanto nombres y convenciones del lenguaje de programación Java.

AJAX (acrónimo de Asynchronous Javascript and XML) es una técnica de desarrollo web para crear aplicaciones interactivas ejecutadas en el lado del cliente. Eso significa que se ejecuta en el navegador mientras se mantiene una comunicación asíncrona con el servidor en segundo plano, lo cual permite hacer cambios en la página web sin necesidad de recargarla, aumentando así la velocidad de la aplicación y mejorando su usabilidad e interactividad.

6.3.2 Visualización de la aplicación Web

El aspecto de la aplicación Web inicial se muestra en la figura 39.
Inicialmente se muestra un mapa vacío y tres botones para realizar tres tareas independientes.

- El botón "Actualizar mapa" ejecuta una función para solicitar los datos de contexto actuales del dispositivo LoRa al Context Broker. Estos datos se representan en el mapa de la aplicación web, centrándolo en las coordenadas en las que se encuentra la entidad.
- El botón "Mostrar histórico" ejecuta una función para solicitar a STH-Comet los datos históricos de contexto. Con ellos se mostrará una tabla con las 10 últimas posiciones, en coordenadas, del dispositivo LoRa.
- El botón "Mostrar alturas" ejecuta una función para solicitar a STH-Comet los datos históricos sobre la altura que ha tomado el dispositivo. Con los datos obtenidos se representa una gráfica.

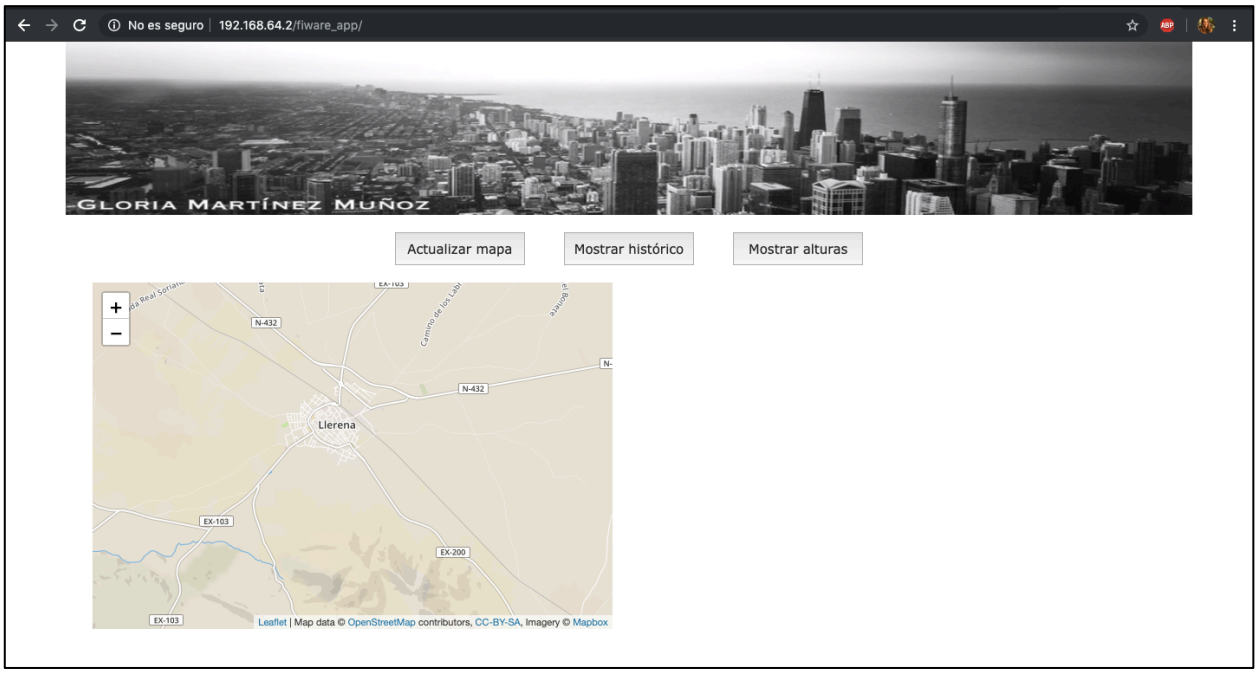

Ilustración 39: Pantalla inicial de la aplicación Web

Cuando se pulsa el botón "Actualizar mapa", la vista del mismo se centra alrededor de la posición obtenida a través del Context Broker, y se representa el punto exacto en el que se encuentra el dispositivo.

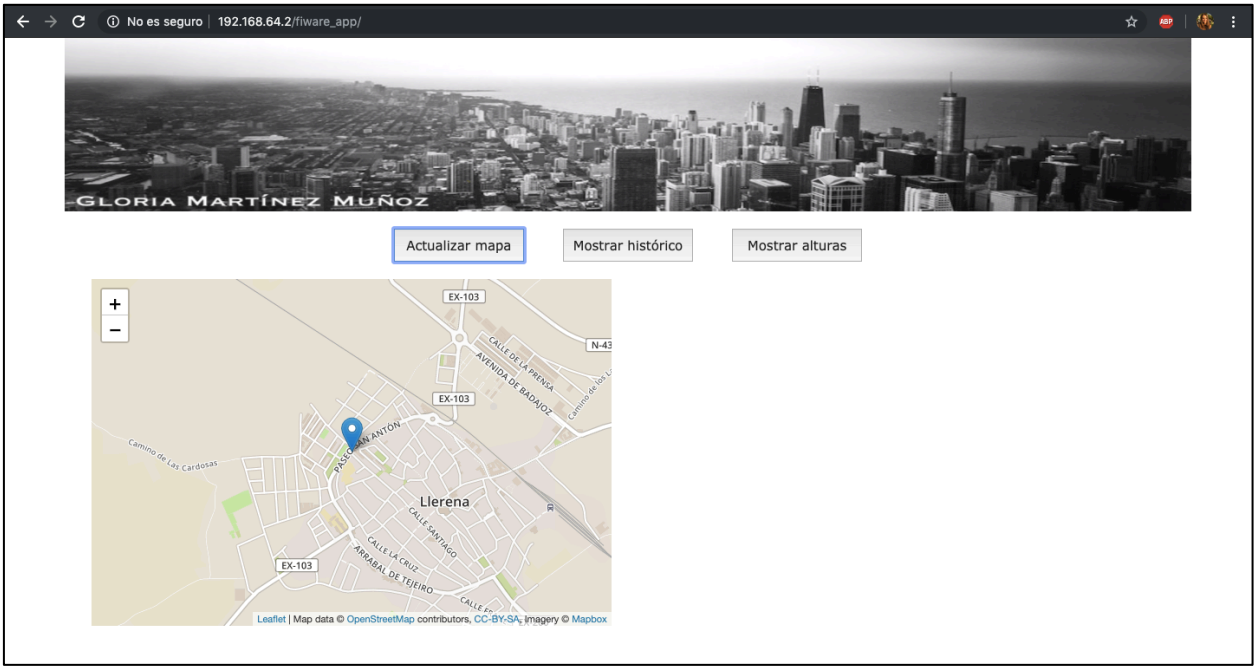

Ilustración 40: Resultado de actualizar el mapa

Además, si se pulsa sobre la ubicación del dispositivo, aparece la altura a la que se encuentra y su ID. Si se pulsa en cualquier otro punto del mapa, se muestran sus coordenadas correspondientes.

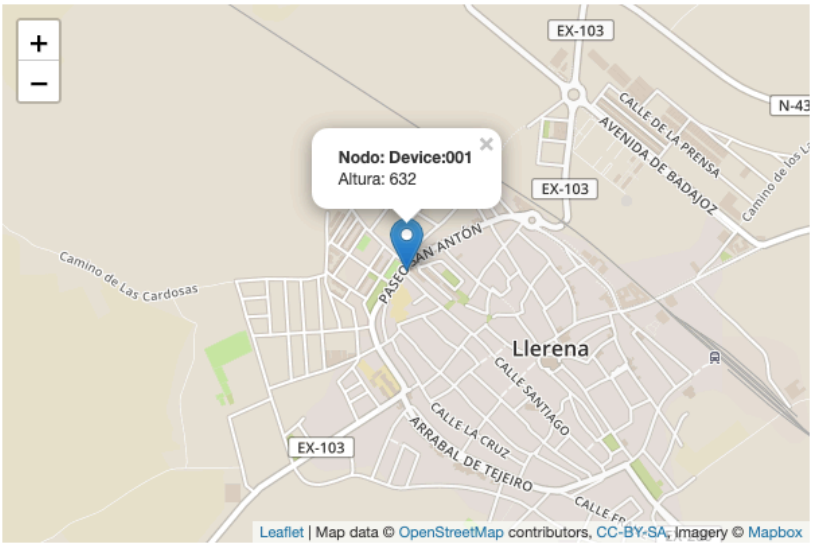

Ilustración 41: Resultado de hacer clic sobre la ubicación del dispositivo

Ilustración 42: Resultado de hacer clic sobre cualquier punto del mapa

Cuando se pulsa el botón "Mostrar histórico" se representa la tabla con los datos históricos de localización y la hora a la que han sido registrados. Además, se pinta sobre el mapa el recorrido realizado por el dispositivo Lora hasta llegar al punto en el que se encuentra en el momento actual.

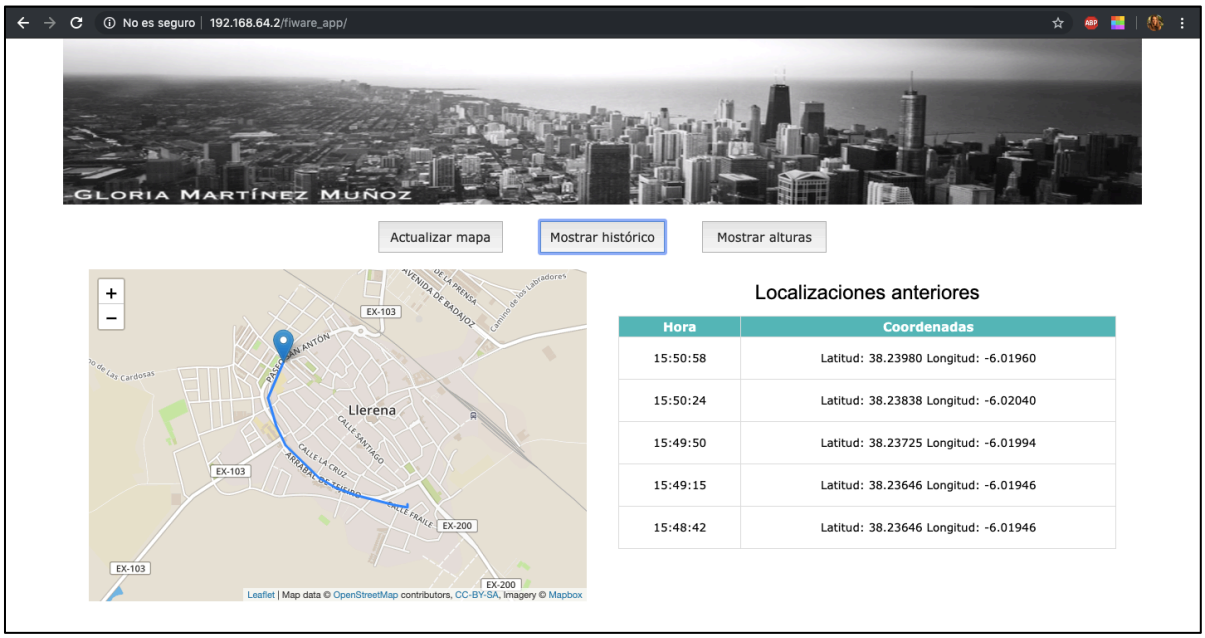

Ilustración 43: Resultado de mostrar el histórico de datos de localización

Cuando se pulsa el botón "Mostrar alturas" se representa la gráfica anteriormente descrita.

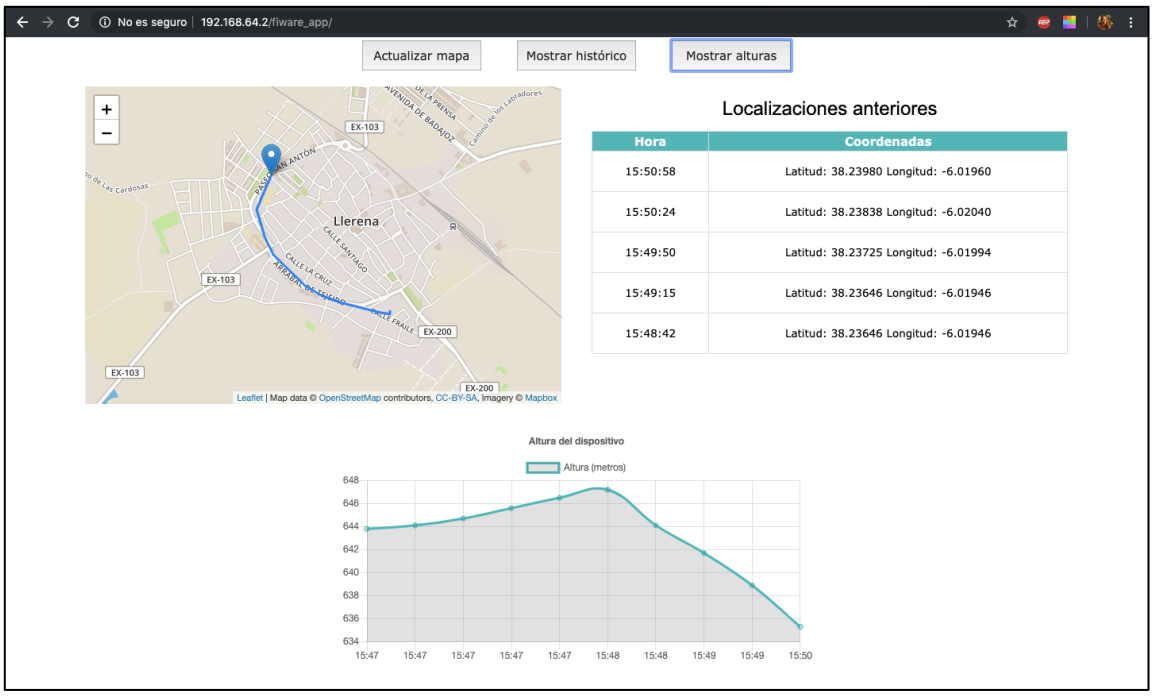

Ilustración 44: Resultado de mostrar la gráfica de alturas del dispositivo

6.3.3 Desarrollo e implementación de la aplicación Web

En esta sección se explican las partes más importantes del código que conforma la aplicación web desarrollada.

Para posibilitar su funcionamiento y desarrollo, se ha empleado el servidor web XAMPP, de código abierto, el cual permite instalar Apache en cualquier sistema operativo. Además, incluye servidores de bases de datos como MySQL o SQLite, aunque no se han empleado en este proyecto puesto que la función de almacenamiento del histórico de contexto ya recae sobre la aplicación Fiware.

Una vez inicializado el servidor de XAMPP, podremos montar un volumen en el que alojar los archivos necesarios para la aplicación web a desarrollar, y acceder a ella mediante la IP que XAMPP proporciona, como puede verse en la figura 45.

Ilustración 45: Ventana principal de XAMPP

6.3.3.1 Conexión con el Context Broker

Para realizar la comunicación entre la aplicación web y el context Broker, se ha empleado PHP para realizar las peticiones GET necesarias.

El siguiente código se corresponde con una query al Context Broker en la que se obtienen las coordenadas actuales del dispositivo, así como su altura.

```
<?php
    $curl = curl_init();
    curl_setopt_array($curl, array(
      CURLOPT_PORT => "1026",
      CURLOPT_URL     =>     "http://192.168.64.1:1026/v2/entities/urn:ngsi-
ld:Device:001?options=values&attrs=location,value",
      CURLOPT_RETURNTRANSFER => true,
      CURLOPT_ENCODING => "",
      CURLOPT_MAXREDIRS => 10,
      CURLOPT_TIMEOUT => 30,
      CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
      CURLOPT_CUSTOMREQUEST => "GET",
      ),
    );

    $response = curl_exec($curl);
    $err = curl_error($curl);

    curl_close($curl);

    if ($err) {
      echo "cURL Error #:" . $err;
        echo $response;
    } else {
      echo $response;
    }

?>
```

En la URL anterior se solicita al Context Broker el valor actual de los atributos `location` y `value` para el dispositivo con id "`urn:ngsi-ld:Device:001`".

Para acceder a los datos históricos del contexto, la query se realiza al puerto 8666, donde escucha el componente STH-Comet. Además, es necesario indicar el servicio al que se hace referencia en la cabecera de la petición. En el caso de esta aplicación fiware no se creó ningún servicio en concreto, por lo que el nombre del servicio a indicar es `fiware-default`. En la URL se especifica que se desean recuperar las últimas 10 muestras del atributo de localización, con un offset = 0.

```
<?php
    $curl = curl_init();

    curl_setopt($curl,CURLOPT_URL,

"http://192.168.64.1:8666/STH/v1/contextEntities/type/Device/id/urn:ngsi-
ld:Device:001/attributes/location?hLimit=10&hOffset=0");
    curl_setopt($curl,CURLOPT_PORT,"8666");
    curl_setopt($curl,CURLOPT_RETURNTRANSFER,true);
```

```
      curl_setopt($curl,CURLOPT_ENCODING,"");
      curl_setopt($curl,CURLOPT_MAXREDIRS,10);
      curl_setopt($curl,CURLOPT_TIMEOUT,10);
      curl_setopt($curl,CURLOPT_HTTP_VERSION,CURL_HTTP_VERSION_1_1);
      curl_setopt($curl,CURLOPT_CUSTOMREQUEST,"GET");

      $headers = [
          'fiware-service: default',
          'fiware-servicepath: /'
      ];

      curl_setopt($curl, CURLOPT_HTTPHEADER, $headers);

      $response = curl_exec($curl);
      $err = curl_error($curl);

      curl_close($curl);

      if ($err) {
        echo "cURL Error #:" . $err;
          echo $response;
      } else {
        echo $response;
      }

  ?>
```

Para realizar la llamada a estas funciones PHP se emplea Javascript. Un ejemplo de llamada es el que se muestra en el siguiente extracto de código:

```
$.ajax({
             type:'GET',
             url: 'get_coords.php',
             async: true,
             success:function(response){

                  //Procesado de la respuesta

                  //Actualización de elementos de la página web

            },
            error:function(response){
                  alert("Ha fallado la conexión con Orion Context Broker");
            }
          });
```

6.3.3.2 Inserción de un mapa

Para poder visualizar el mapa de la web se ha utilizado la librería Leaflet [22] de código abierto. En un principio se intentó realizar esta función utilizando la API de Google Maps, pero su uso actualmente es de pago y existen otras muchas alternativas bastante potentes.

El siguiente extracto de código hace referencia a la introducción del mapa en la web, si bien es cierto que en otras partes del código se realizan funciones para actualizar el mapa o introducir otros elementos en él.

```
//Se añade el mapa y se establecen las coordenadas iniciales y el zoom
var mymap = L.map('mapid').setView([38.233, -6.012], 13);

L.tileLayer('https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}.png?access_tok
en=pk.eyJ1IjoibWFwYm94IiwiYSI6ImNpejY4NXVycTA2emYycXBndHRqcmZ3N3gifQ.rJcFIG2
14AriISLbB6B5aw', {
                maxZoom: 18,//Zoom maximo permitido para el mapa
                attribution: 'Map data © <a
href="https://www.openstreetmap.org/">OpenStreetMap</a> contributors, ' +
                        '<a href="https://creativecommons.org/licenses/by-
sa/2.0/">CC-BY-SA</a>, ' +
                        'Imagery © <a
href="https://www.mapbox.com/">Mapbox</a>',
                id: 'mapbox.streets'
        }).addTo(mymap);

// Se inicializa el marcador que muestra la posicion del dispositivo
var marcador = L.marker([0,0]).addTo(mymap)

// Se inicializa el elemento popup para mostrar un mensaje cuando se pinche
sobre el dispositivo
var popup = L.popup();
function onMapClick(e) {
    popup
        .setLatLng(e.latlng)
        .setContent("Has pinchado las coordenadas: " + e.latlng.toString())
        .openOn(mymap);
        }

// Cuando se pinche en cualquier otro punto del mapa, se mostraran sus
coordenadas
mymap.on('click', onMapClick);

// Se inicializa el elemento para dibujar la trayectoria del dispositivo
var polyline = L.polyline([0,0]).addTo(mymap);
```

6.3.3.3 Inserción de un gráfico

Para añadir la gráfica de altura que muestra la aplicación web, se ha usado la librería Chart.js [23].

En el siguiente extracto de código se muestra la definición e inserción de la gráfica, tomando como datos las `alturas` obtenidas a través de una petición a STH-Comet y las `horas` a las que se han tomado dichas muestras.

```
var chart = new Chart(ctx, {
                type: 'line',
                data:{
                    datasets: [{
                    data: alturas, // Array con histórico de alturas
                    borderColor: '#2eb8b8',
                    fill:true,
                    label: 'Altura (metros)'}],

                    labels: horas // Array con horas de muestras
                 },
                 options: {
                    title: {
                    display: true,
```

```
                    text: 'Altura del dispositivo'
                    }
                }
                });
```

7 PRUEBAS Y VALIDACIONES

En este capítulo se exponen las pruebas realizadas con los dispositivos de la red de sensores LoRa. Cabe destacar que tras trabajar con ellos se ha podido comprobar que no eran de gran calidad. Se han encontrado ciertos problemas durante su configuración y algunos aspectos de mal funcionamiento han impedido realizar algunas pruebas que habrían sido interesantes.

7.1 Pruebas LoRa

Para poner a prueba esta tecnología se han realizado varios experimentos para probar el alcance de la misma, detallados en esta sección.

El alcance de los enlaces que pueden establecerse con esta tecnología no está muy claro, ya que depende de números factores:

- El ruido electromagnético en el lugar de la medición.
- El tipo y tamaño de la antena empleada.
- La configuración de potencia en el receptor.
- La posición y altura de los nodos.
- Los obstáculos entre los nodos.

Según la teoría, la tecnología LoRa permite establecer enlaces entre nodos de hasta 15Km, con una tasa de datos que van desde los 300 hasta los 50.000 por segundo.

7.1.1 Pruebas de alcance en ciudad

Se ha realizado una prueba de alcance dentro del núcleo Urbano de Sevilla, para comprobar hasta qué punto la tecnología LoRa es capaz de funcionar con obstáculos. Esta prueba pretendía probar sobre todo este tema y no la distancia, ya que, en los grandes núcleos urbanos, la presencia de altos edificos puede complicar las comunicaciones entre nodos y es necesario conocer las capacidades de la tecnología.

En la figura 46 se muestra la imagen del mapa de Sevilla obtenida por Google Maps y los dos puntos que forman los extremos de la comunicación establecida. El nodo receptor, en el punto A, se encuentra situado en la azotea de un edificio de 55 metros, uno de los más altos de la zona.

El nodo transmisor en el punto B, se ha desplazado desde el punto A hasta lograr distanciarse una distancia considerable como para comprobar que los edificios encontrados por el camino no eran suficiente para frenar la señal LoRa. El punto B señalado en el mapa no indica la máxima distancia del enlace, ya que la

señal en este punto se seguía recibiendo sin problema alguno. Sin embargo, resultaba complicado distanciar los nodos más entre sí ya que para la realización de la prueba se necesitaba ayuda de otra persona que dispusiera del nodo transmisor.

Ilustración 46: Enlace establecido dentro de un núcleo urbano

Como muestra la imagen satélite de Google Maps, la distancia entre ambos puntos en línea recta es de 990m aproximadamente. Teniendo en cuenta la altura a la que se situaba el nodo transmisor, la distancia real del enlace resulta:

$$distMapa^2 + altura^2 = distReal^2$$

$$distReal = \sqrt{distMapa^2 + altura^2} = \sqrt{990m^2 + 55m^2} = 991.52\ metros$$

7.1.2 Pruebas de alcance en espacio más abierto

Para comprobar la máxima distancia a la que los nodos utilizados eran capaces de establecer conexión, se ha realizado otra prueba en un pueblo, en el que la ausencia de altos edificos y el espacio más despejado permiten una mejor comunicación entre los nodos y el establecimiento de un enlace con mayor alcance. Los resultados han sido bastante satisfactorios.

En la figura 47 se muestra la vista aérea de la población de Llerena, en Badajoz. Sobre ella se ha trazado el recorrido realizado por el dispositivo transmisor. El nodo receptor se encuentra situado en el punto A, en la terraza de una vivienda. El nodo transmisor se ha desplazado a lo largo de toda la circunvalación del pueblo, y en todo momento la señal se ha recepcionado satisfactoriamente.

Ilustración 47: Recorrido realizado por el dispositivo transmisor

La máxima distancia alcanzada se señala en el siguiente mapa en el punto B.

Ilustración 48: Enlace más distante establecido en la población de Llerena

Como puede comprobarse, en enlace ha permanecido establecido atravesando toda la población y alcanzando un punto situado a las afueras de ésta.

En la imagen 49 se muestra la vista desde la terraza de la vivienda en la que se situaba el nodo receptor. Con ello se pretende dejar una evidencia de que, a pesar de no haber altos edificios, tampoco había una visión totalmente directa entre ambos dispositivos.

Ilustración 49: Vista de la población desde la ubicación del dispositivo receptor

7.1.3 Prueba de alcance en interiores

Por último, se ha comprobado que el enlace entre los nodos se mantiene a pesar de permanecer uno de ellos en el interior de un edifico. Concretamente, mientras se realizaban las pruebas, el nodo transmisor permanecía en el interior del edificio, incluso dentro de un ascensor, y la señal seguía siendo recibida por el nodo receptor, situado en el exterior.

También se ha realizado la prueba tapando la antena de los dispositivos con la mano, y la señal se ha seguido recibiendo. Obviamente, el alcance de la comunicación en estas condiciones, desciende bastante.

7.1.4 Comparación con la tecnología Zigbee

En este apartado se hace referencia a otro proyecto, que realicé como Trabajo de Fin de grado. En él se diseñaba una red de sensores que utilizaban la tecnología Zigbee. El propósito de esta sección es comparar las pruebas de alcance obtenidas en cada uno de los proyectos.

La siguiente figura está extraída de dicho Trabajo de Fin de Grado. Se trata de una vista aérea de la Escuela Técnica Superior de Ingeniería de Sevilla.

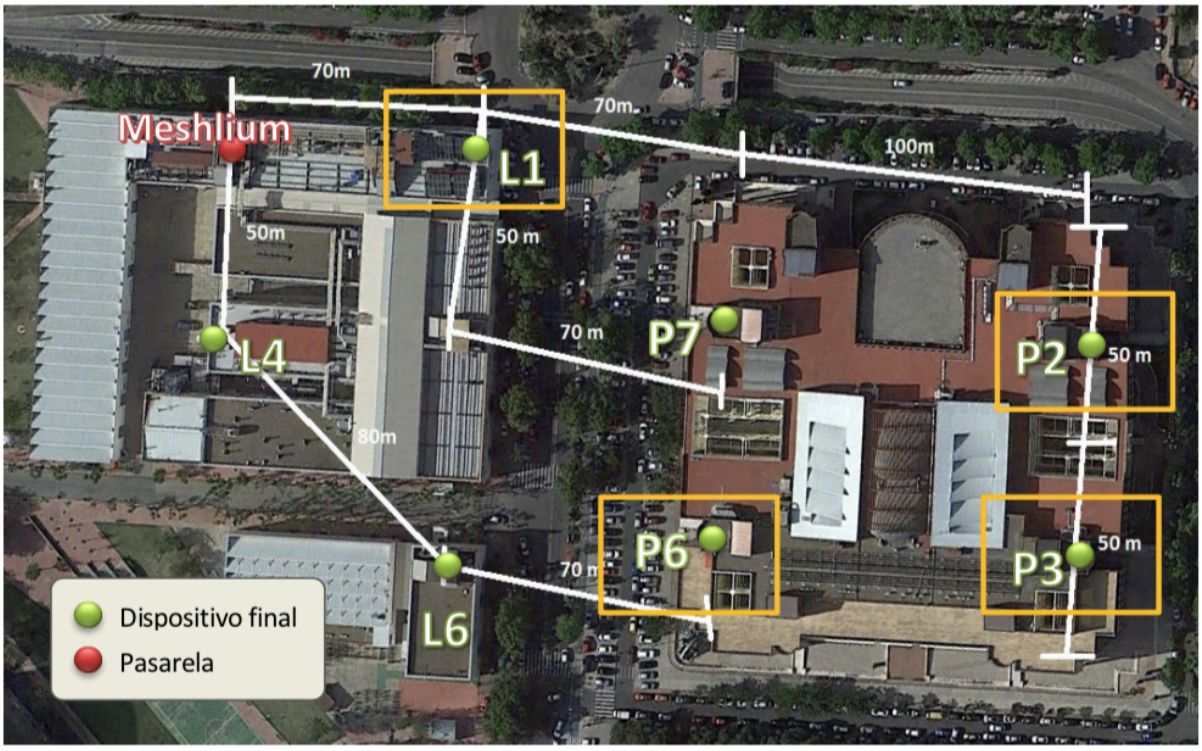

Ilustración 50: Imagen extraída de proyecto Zigbee

En la imagen aparecen, en color verde, los puntos en los que fueron situados los nodos Zigbee y, en rojo, el punto donde se situaba la pasarela Zigbee que recolectaba los datos recibidos por cada uno de los nodos sensores.

Los cuatro puntos que aparecen remarcados (L1, P2, P3 y P6), corresponden a los nodos que, debido a las distancias entre enlaces o a obstáculos que impedían la visibilidad directa, carecían de conectividad con la pasarela.

Como puede apreciarse, en ningún caso las distancias establecidas entre los nodos eran superiores a 250 metros. Incluso en el peor de los casos, la distancia entre el nodo L1 y la pasarela era de unos escasos 70 metros.

Con esta comparativa queda remarcado el gran potencial de LoRa respecto a Zigbee en cuanto al alcance de los enlaces establecidos.

7.1.5 Pruebas de consumo

Se ha intentado realizar alguna prueba de consumo de los dispositivos, a fin de comprobar que los valores obtenidos fueran razonables y se adecuaran a uno de los principales requisitos de una red de sensores: el bajo consumo de energía.

Esta prueba ha resultado imposible de realizar, por dos motivos principales:

1. Los dispositivos Heltec Wifi Lora 32 empleados en este proyecto no disponen de ninguna función para obtener el nivel de batería actual del mismo. A pesar de no poder obtener este valor directamente mediante programación, sería posible obtener una medida del voltaje del dispositivo en funcionamiento y realizar un cálculo para hallar la previsión de consumo en un cierto intervalo de tiempo. Sin embargo, se ha encontrado otro problema.

2. A pesar de haber dotado al dispositivo transmisor de una batería que cumpliera las especificaciones de uso del nodo, ha resultado imposible lograr que el aparato se encendiera cuando su única fuente de alimentación era dicha batería. Buscando información sobre el tema se han encontrado numerosos usuarios con el mismo problema, por lo que se ha llegado a la conclusión de que el problema no era de programación o de incompatibilidad con la batería utilizada.

Habría resultado muy interesante realizar varias pruebas de consumo con diferentes modos de funcionamiento de los dispositivos, incluyendo funciones de hibernación del nodo transmisor cuando no tuviera que realizar ninguna tarea.

8 CONCLUSIONES Y LÍNEAS FUTURAS

8.1 Conclusiones

Realizar este proyecto ha sido bastante enriquecedor para mí, tanto como para mi conocimiento como para aumentar mis capacidades para el futuro laboral, ya que las tecnologías utilizadas están en pleno apogeo y son muchas las compañías y empresas que se dedican a las plataformas de redes de sensores y su uso en el Big Data, así como el Internet de las Cosas

Durante la realización del mismo se han tenido que aplicar conocimientos adquiridos durante mis años de formación tanto en el grado como en el máster, y también se han adquirido muchos otros nuevos sobre materias que no cubre el proyecto docente y que eran desconocidos para mí antes de comenzar este proyecto.

A pesar de que en un principio los dispositivos empleados para diseñar la red de sensores parecían muy sencillos de programar, se han encontrado algunos problemas a la hora de hacerlo. Las librerías que implementan han sido modificadas porque fallaban en muchas ocasiones; los métodos definidos en ella son métodos sobreescritos a partir de otros ya existentes en las librerías de Arduino, por lo que su definición causaba problemas de funcionamiento.

Además, el gran problema de la batería ha imposibilitado la realización de otro tipo de pruebas.

8.2 Líneas futuras

Este proyecto tiene un gran margen de mejora, y hay muchas posibilidades de amplicación. El trabajo realizado no es más que una base de la que partir con la que pueden realizarse grandes proyectos. Como mejoras al trabajo realizado, se plantean varias posibilidades:

- Adición de más nodos a la red, ya que la arquitectura actual es bastante simple y está destinada sobre todo a la experimentación de las posibilidades que ofrecen las tecnologías utilizadas. En una aplicación real sería conveniente disponer de más sensores.
- Adición de funcionalidades a la aplicación web.
- Creación de una aplicación móvil. Hoy en día resulta mucho más útil este tipo de formato que una aplicación web.
- Utilización de más elementos Fiware para añadir más funcionalidad a la aplicación.

REFERENCIAS

[1] S. C. Mukhopadhyay, Internet Of Things, 2014.

[2] «Sigfox,» [En línea]. Available: https://www.sigfox.es/.

[3] «Narrowband IoT,» [En línea]. Available: https://accent-systems.com/es/nb-iot/.

[4] «BLE,» [En línea]. Available: https://es.wikipedia.org/wiki/Bluetooth_de_baja_energ%C3%ADa.

[5] «Zigbee,» [En línea]. Available: https://www.zigbee.org/.

[6] «Semtech,» [En línea]. Available: https://www.semtech.com/.

[7] «Lora Alliance,» [En línea]. Available: https://lora-alliance.org/.

[8] «Fiware,» [En línea]. Available: https://www.fiware.org/.

[9] «Placas Heltec WiFi LoRa 32,» [En línea]. Available: https://heltec.org/project/wifi-lora-32/.

[10] «Módulo GPS NEO 6M,» [En línea]. Available: https://randomnerdtutorials.com/guide-to-neo-6m-gps-module-with-arduino/.

[11] «U-Blox,» [En línea]. Available: https://www.u-blox.com/en.

[12] «Cygnus,» [En línea]. Available: https://fiware-cygnus.readthedocs.io/en/latest/index.html.

[13] «Apache Flume,» [En línea]. Available: https://flume.apache.org/.

[14] «STH - Comet,» [En línea]. Available: https://fiware-sth-comet.readthedocs.io/en/latest/index.html.

[15] «Arduino,» [En línea]. Available: https://www.arduino.cc/.

[16] «Docker,» [En línea]. Available: https://www.docker.com/.

[17] «cURL,» [En línea]. Available: https://curl.haxx.se/.

[18] «GEO Json,» [En línea]. Available: https://geojson.org/.

[19] «FIWARE Lab,» [En línea]. Available: https://cloud.lab.fiware.org/auth/login/?next=/.

[20] «HTML y CSS,» [En línea]. Available: https://www.w3.org/standards/webdesign/htmlcss.

[21] «World Wide Web Consorcium,» [En línea]. Available: https://www.w3.org/.

[22] «Leaflet,» [En línea]. Available: https://leafletjs.com/.

[23] «Chart.js,» [En línea]. Available: https://www.chartjs.org/.

ANEXO 1: INSTALACIÓN DE DOCKER

Para llevar a cabo la instalación de Docker en el servidor CentOS utilizado, es necesario seguir los siguientes pasos:

Primeramente, se instalan una serie de paquetes necesarios:

```
$ sudo yum install -y yum-utils device-mapper-persistent-data lvm2
```

Se configura el repositorio de Docker-ce y posteriormente se instala:

```
$ sudo yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo

$ sudo yum install docker-ce
```

Se configura Docker para que se inicie al inicio del servidor CentOS:

```
$ sudo systemctl enable docker.service
```

Se inicia el servicio Docker:

```
$ sudo systemctl start docker.service
```

Además de la instalación de Docker, también se ha realizado la de Docker-compose, una herramienta que facilita la definición y ejecución de aplicaciones con múltiples contenedores, como es el caso de este proyecto. Para instalarla, se siguen los siguientes pasos:

Se instalan paquetes extra necesarios:

```
$ sudo yum install epel-release
```

Se instala la herramienta Python PIP:

```
$ sudo yum install -y python-pip
```

Se instala Docker-compose:

```
$ sudo pip install docker-compose
```

A continuación, se muestra la lista de comandos a ejecutar con Docker:

```
 Options:
       --config string      Location of client config files (default
                            "/Users/gloria/.docker")
   -D, --debug              Enable debug mode
   -H, --host list          Daemon socket(s) to connect to
   -l, --log-level string   Set the logging level
                            ("debug"|"info"|"warn"|"error"|"fatal")
                            (default "info")
       --tls                Use TLS; implied by --tlsverify
       --tlscacert string   Trust certs signed only by this CA (default
                            "/Users/gloria/.docker/ca.pem")
       --tlscert string     Path to TLS certificate file (default
                            "/Users/gloria/.docker/cert.pem")
       --tlskey string      Path to TLS key file (default
                            "/Users/gloria/.docker/key.pem")
       --tlsverify          Use TLS and verify the remote
   -v, --version            Print version information and quit

 Management Commands:
   builder     Manage builds
   config      Manage Docker configs
   container   Manage containers
   image       Manage images
   network     Manage networks
   node        Manage Swarm nodes
   plugin      Manage plugins
   secret      Manage Docker secrets
   service     Manage services
   stack       Manage Docker stacks
   swarm       Manage Swarm
   system      Manage Docker
   trust       Manage trust on Docker images
   volume      Manage volumes

 Commands:
   attach       Attach local standard input, output, and error streams
to a running container
   build       Build an image from a Dockerfile
   commit      Create a new image from a container's changes
   cp            Copy files/folders between a container and the local
filesystem
   create      Create a new container
   diff        Inspect changes to files or directories on a container's
filesystem
   events      Get real time events from the server
   exec        Run a command in a running container
   export      Export a container's filesystem as a tar archive
   history     Show the history of an image
   images      List images
   import     Import the contents from a tarball to create a filesystem
image
   info        Display system-wide information
```

```
    inspect     Return low-level information on Docker objects
    kill        Kill one or more running containers
    load        Load an image from a tar archive or STDIN
    login       Log in to a Docker registry
    logout      Log out from a Docker registry
    logs        Fetch the logs of a container
    pause       Pause all processes within one or more containers
    port       List port mappings or a specific mapping for the container
    ps          List containers
    pull        Pull an image or a repository from a registry
    push        Push an image or a repository to a registry
    rename      Rename a container
    restart     Restart one or more containers
    rm          Remove one or more containers
    rmi         Remove one or more images
    run         Run a command in a new container
    save         Save one or more images to a tar archive (streamed to
STDOUT by default)
    search      Search the Docker Hub for images
    start       Start one or more stopped containers
    stats        Display a live stream of container(s) resource usage
statistics
    stop        Stop one or more running containers
    tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
    top         Display the running processes of a container
    unpause     Unpause all processes within one or more containers
    update      Update configuration of one or more containers
    version     Show the Docker version information
    wait       Block until one or more containers stop, then print their
exit codes
```

ANEXO 2: CONFIGURACIÓN DE FIWARE LAB

En este anexo se recogen los pasos básicos para la creación de una máquina virtual en Fiware Lab.

Al iniciar sesión en la plataforma de FIWARE Lab con la cuenta correspondiente, el usuario encuentra el siguiente panel de control, donde pueden verse los recursos utilizados y disponibles según el tipo de cuenta:

Ilustración 51: Panel principal FIWARE Lab

El primer paso para crear una máquina virtual es crear un **par de claves** que nos permitirán posteriormente acceder a los servicios. Para ello se debe accedera la sección "Par de claves" dentro de "Acceso y Seguridad". Al elegir la opción "+ Crear par de claves" aparecerá la pantalla que se muestra en la figura 52. En el diálogo se debe introducir el nombre deseado para el par de claves a crear y aceptar, tras lo cual se descargará un fichero con las claves, necesario para conectar a la máquina virtual por SSH.

Ilustración 52: Creación y descarga de un par de claves

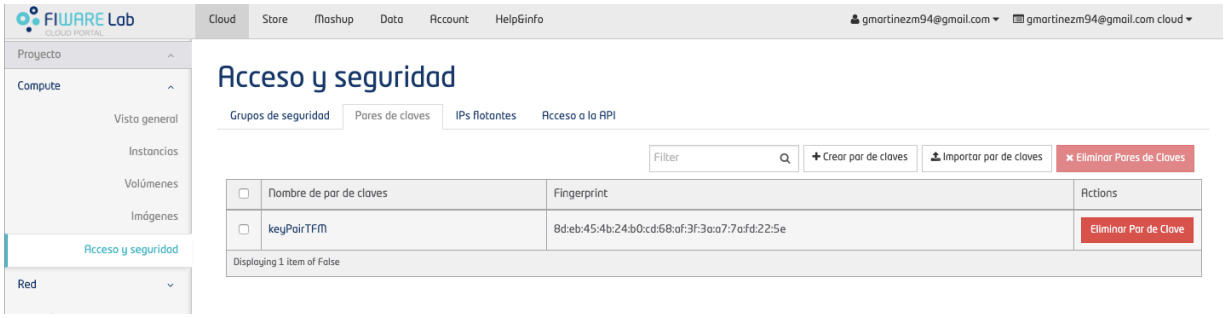

Ilustración 53: Par de claves creado

El siguiente paso es definir un **grupo de seguridad** que aplicaremos posteriormente a la máquina virtual. En este grupo de seguridad se definen los puertos que la máquina tiene abiertos y las direcciones o grupos de direcciones IP que tendrán acceso a ella. Existe un grupo por defecto ya creado, que puede emplearse en el caso de no necesitar restringir el acceso a niguna dirección IP. Pueden añadirse nuevas reglas al grupo por defecto o bien crear grupos nuevos.

Para crear un grupo de seguridad nuevo, en caso de necesitarlo, en la sección "Acceso y Seguridad" se debe seleccionar la pestaña "Grupos de seguridad". En la esquina superior derecha se debe seleccionar "+ Crear grupo de seguridad", tras lo cual aparecerá un diálogo como el que se muestra en la figura 54, en el que se debe definir un nombre para el grupo y su descripción.

Ilustración 54: Creación de un grupo de seguridad

Tras su creación, el nuevo grupo aparecerá listado junto al grupo por defecto.

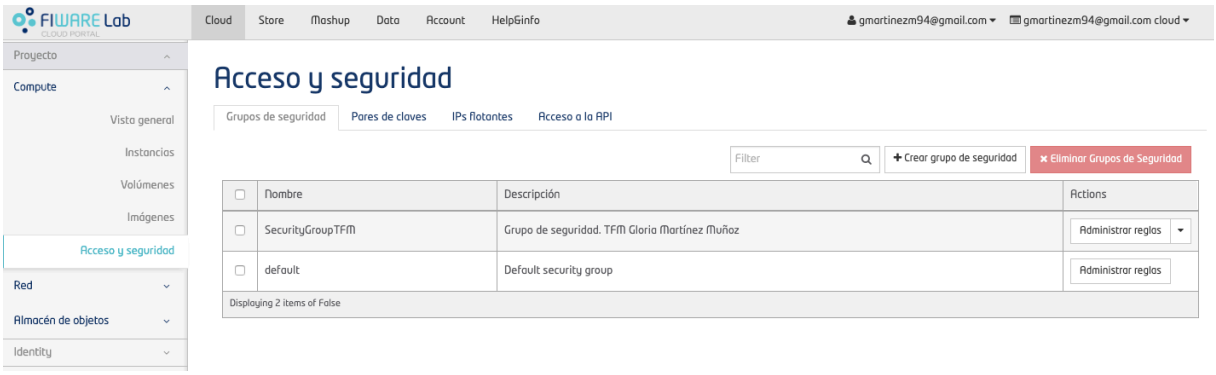

Ilustración 55: Listado de grupos de seguridad

Para añadir reglas al grupo, se debe seleccionar la opción "Administrar reglas" del grupo deseado. Aparecerá un diálogo como el que se muestra en la figura 56, donde será necesario escoger el tipo de regla a crear (por defecto vienen las más típicas ya configuradas, aunque es posible configurar otros puertos para diseñar reglas a medida).

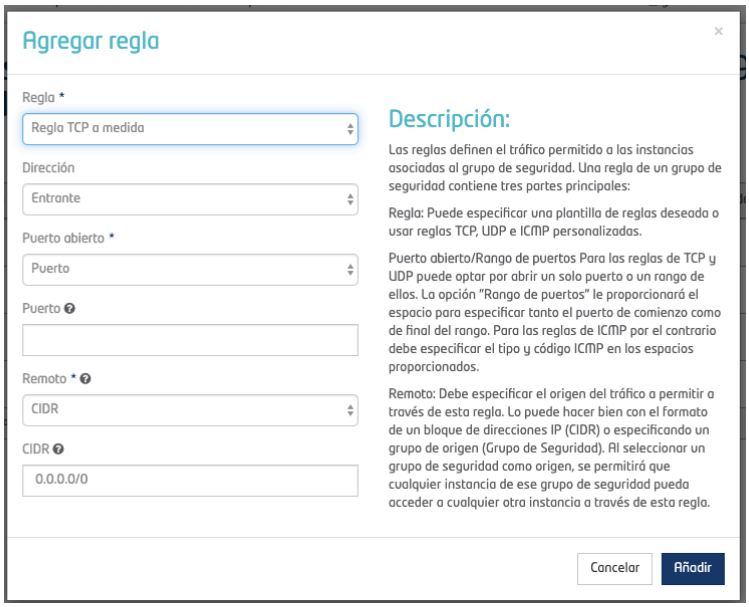

Ilustración 56: Configuración de una nueva regla de seguridad

Tras crear las reglas necesarias, el grupo de seguridad creado se ve como muestra la figura 57.

Ilustración 57: Reglas creadas dentro de un grupo de seguridad

Tanto las reglas como el propio grupo de seguridad pueden modificarse y eliminarse en caso de ser necesario.

El siguiente paso es crear una red a la que conectar la máquina virtual. En la sección "Redes" se debe seleccionar la opción "+ Crear red", tras lo que aparece la siguiente pantalla:

Ilustración 58: Creación de una red (I)

Se debe elegir una dirección para la subred y un normbre, así como la dirección IP asignada a la puerta de enlace.

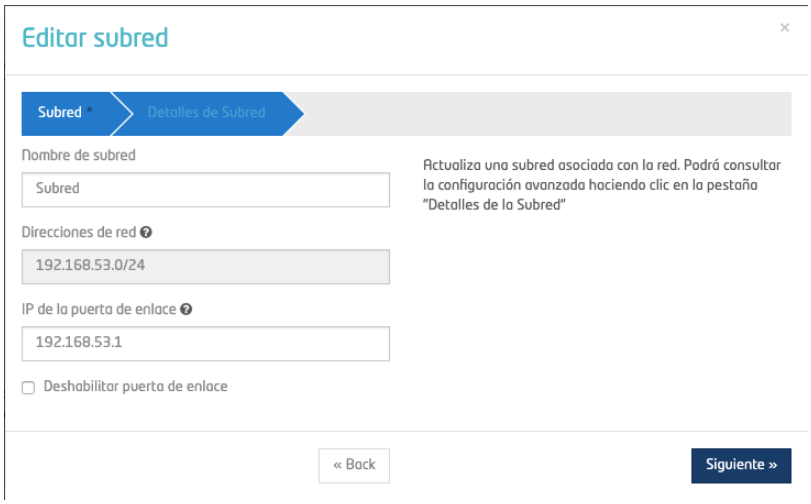

Ilustración 59: Creación de una red (II)

Como último paso para la creación de la red, se configuran los datos relativos a DHCP y DNS.

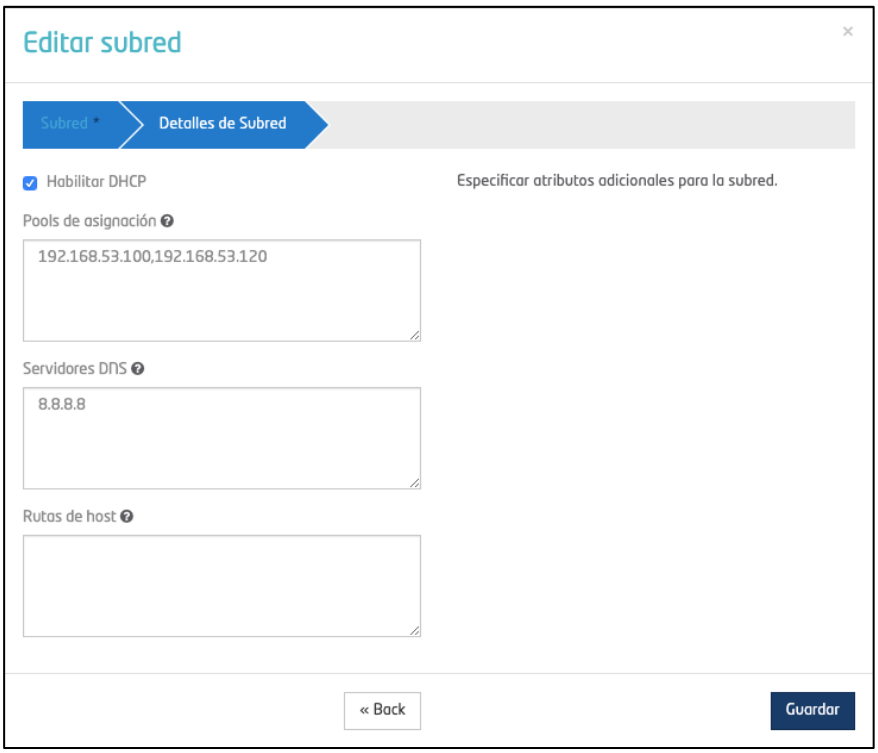

Ilustración 60: Creación de una red (III)

El resultado de la creación se muestra en la siguiente imagen, donde se ven la red recientemente creada y la red por defecto.

Ilustración 61: Redes creadas

Por último, se procede a la **creación de la máquina virtual**. Para ello, en el menú de "Instancias", se debe seleccionar la opción "Lanzar instancia" en la esquina superior derecha, tras lo cual aparece el diálogo mostrado en la figura 62.

En la pestaña "detalles" se debe seleccionar la zona en la que va a crearse la instancia, el nombre, el sabor (equivale al tamaño de la máquina virtual) y la imagen que va a cargarse en su creación. En este caso se ha escogido el sistema operativo Centos 7.

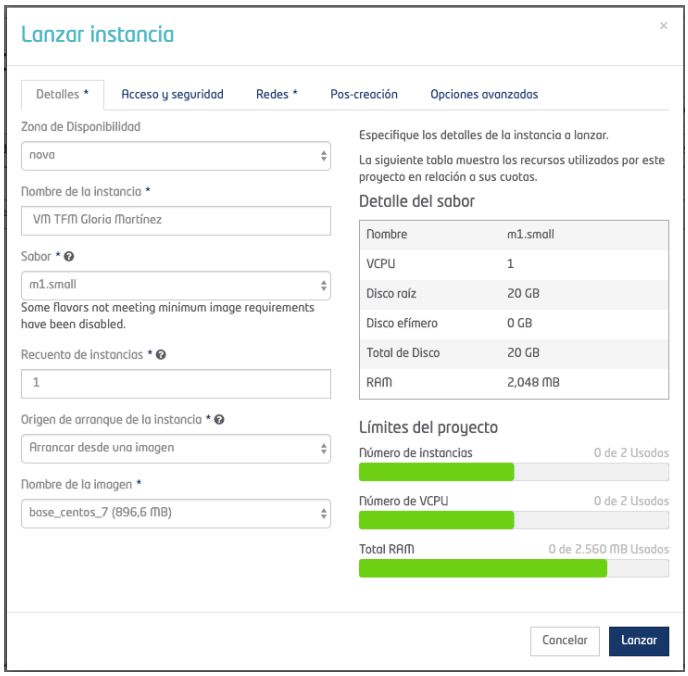

Ilustración 62: Creación de una máquina virtual (I)

En la pestaña de "acceso y seguridad" se debe seleccionar el par de claves generado en el primer paso y el grupo de seguridad que se va a aplicar.

Ilustración 63: Creación de una máquina virtual (II)

En la pestaña de "redes" se debe seleccionar la red a la que va a conectarse la máquina virtual. Por defecto, ya se encuentra una creada. En este caso se ha creado una nueva en el paso anterior, ya que en la red por defecto no se disponen de permisos de administrador.

Ilustración 64: Creación de una máquina virtual (III)

Tras este paso ya puede lanzarse la máquina virtual, tras lo cual se muestra la siguiente pantalla:

Ilustración 65: Creación de una máquina virtual (IV)

Para que nuestra máquina virtual tenga acceso a Internet, podemos **crear un router** y conectar la red de nuestra máquina virtual a la red pública. Para ello, se debe seleccionar la opción "+ Crear router" en el apartado "Routers". Para su configuración únicamente es necesario asignarle un nombre y elegir la red a la que estará conectado por una de sus interfaces, en este caso la red pública.

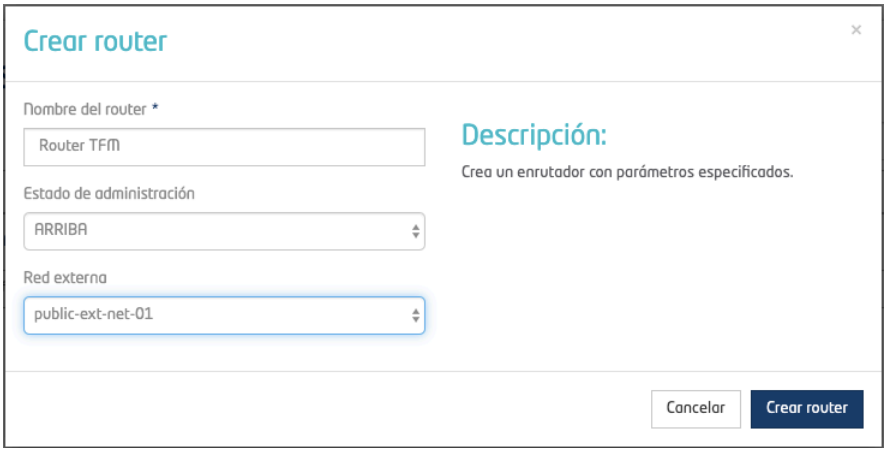

Ilustración 66: Creación de un router

Se puede **asociar una IP flotante a la máquina virtual**. Para ello en el menú “Instancias”, en las opciones de la máquina se escoge “Asignar IP flotante”.

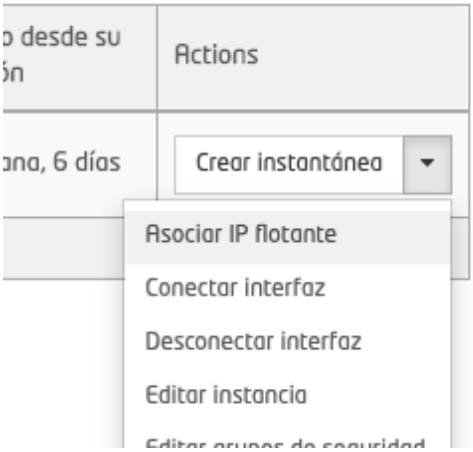

Ilustración 67: Asignar IP flotante a máquina virtual

Una vez completado este paso aparecerá la información de la máquina con la IP flotante asignada.

Ilustración 68: Máquina virtual con IP flotante asignada

Si se accede al apartado “Topología de red” puede verse el esquema de red de nuestra nube en Fiware Lab. En la situación actual aparecerá la máquina virtual conectada a la red que se ha creado, por un lado, el router conectado a la red pública por otro, y la red creada por defecto por otro.

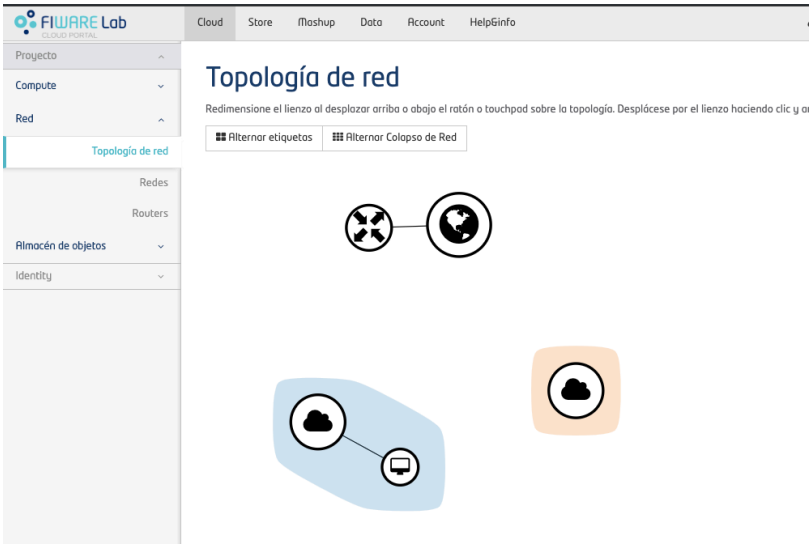

Ilustración 69: Topología de red sin conectar a red pública

Para que la máquina virtual tenga acceso a la red pública es necesario conectar la subred a una de las interfaces del router. Para ello, pinchando sobre el router, se debe escoger la opción "Add interface" y rellenar los datos necesarios para conectar dicha interfaz a nuestra subred.

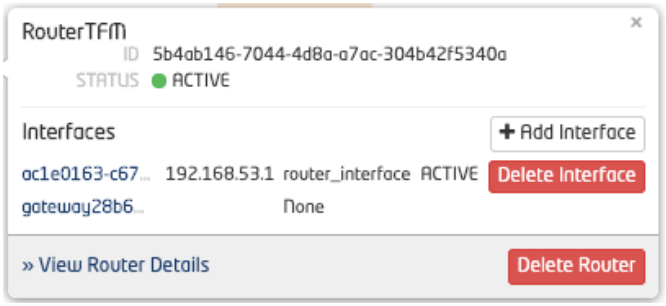

Ilustración 70: Añadir interfaz a router

Tras este paso, la topología queda como se muestra en la siguiente imagen. La máquina virtual ya puede ser accedida desde la red pública.

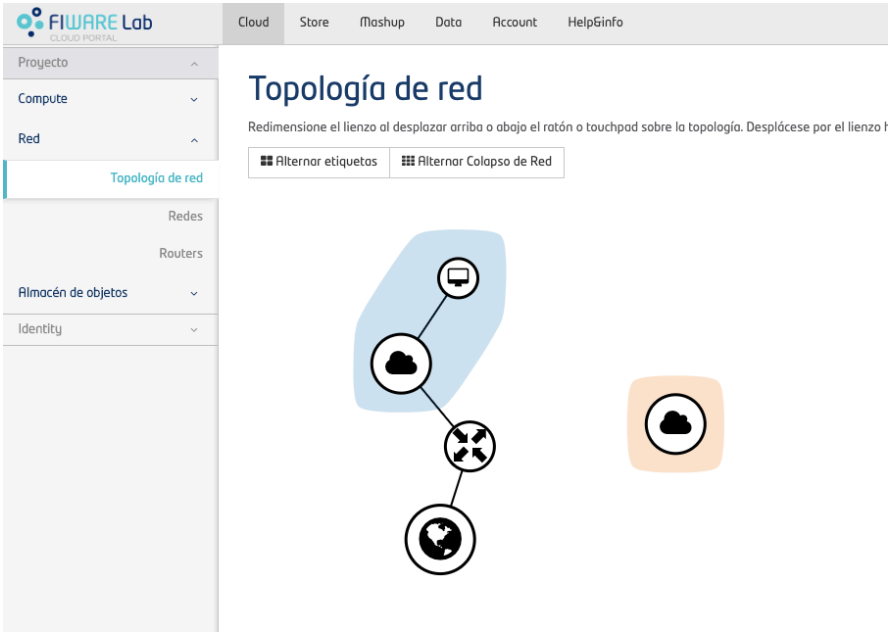

Ilustración 71: Topología de red conectada a red pública

Para el resto de configuraciones y para comenzar a usar la máquina virtual, puede ser accedida mediante SSH utilizando el par de claves creado en los primeros pasos.